IBM

Operating Logic of the 705

INTRODUCTION

The purpose of this manual is to provide a simplified version of 705 machine logic. A knowledge of these principles will aid in understanding more fully the functions that each instruction causes the machine to perform. This knowledge, it is hoped, will also stimulate ideas for better programming which will both reduce machine time and conserve memory space.

As this manual does not always repeat explanations under all headings to which they may pertain, it should not be used as a reference manual until it has been read completely.

This manual assumes a knowledge of 705 programming and familiarity with the "Preliminary Manual of Operations, Type 705", Form No. 22-6627 and subsequent revisions, and with the "IBM 705 Console Manual", Form No. 32-7077.

This manual is the result of work contributed by:

T. E. BRADSHAW, JR.

J. J. HUGHES

TABLE OF CONTENTS

GENERAL

Information in the 705 is represented by electronic signals. These may be regarded in any one of several ways; for example, as either 0 or 1, Yes or No, Bit or No-bit, Up or Down. In this manual the memory cores will be said to be either in a "Bit" or "No-bit" condition, and the logical circuits will be said to be either "Up" or "Down". "Up" or "Down" is used in the logical circuitry as this most closely corresponds to the actual conditions. An Up condition represents a voltage level of + 10 volts; a Down condition represents a voltage level of -30 volts.

Within the 705 there are four basic types of circuits:

1. Triggers

   These are sometimes called "flip-flops". A trigger is an electronic device which can remember two states: On or Off. When a trigger is in the On state, certain of its output lines will be in an Up condition and other output lines will be in a Down condition. If the trigger is flipped to the Off state, then the output lines which previously were in an Up condition will be switched to a Down condition, and the output lines which were in a Down condition will be switched to an Up condition. A trigger can be turned on or off in less than one half of a microsecond.

2. Inverters

   These convert an Up condition to a Down condition, and vice versa. It might be noted at this point that the 705 works quite often on "not" conditions, i.e. the absence of certain conditions. For example, the blank, hyphen, and ampersand are recognized in the 705 by a combination of "not 8","not 4", "not 2", and "not 1" conditions.

3. And Gates

   An And Gate is an electrical circuit which produces an Up condition on its output if all of its input lines are in an Up condition.

4. Or Gates

   This is an electrical circuit which requires that only one of its input lines be Up in order to have its output in an Up condition.

The symbols shown in Figure I will be used in subsequent illustrations.

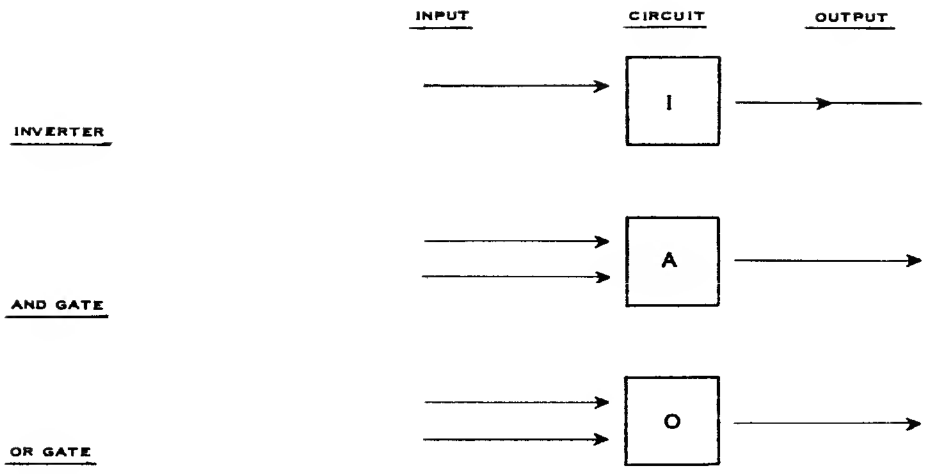


FIGURE 1

Gates and inverters with various inputs and outputs to illustrate their operation are shown in Figure 2.

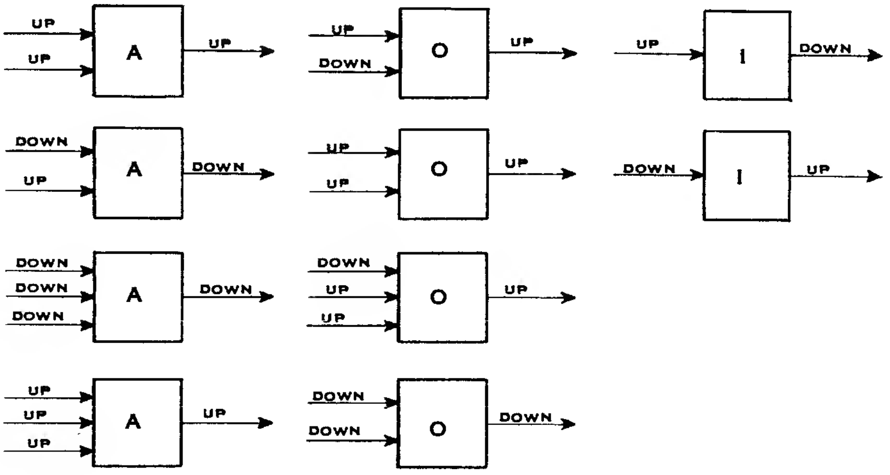


FIGURE 2

These gates, in combination, may be used to execute various logical functions, e.g., recognition of a zero or comma in CR1, as in the SPR instruction. The logical gates for this recognition are shown in Figure 3 as they actually exist in the 705. In this figure, a bar over a numerical or alphabetic character, $(\overline{4})$, represents the "not" condition or the absence of that particular condition or bit.

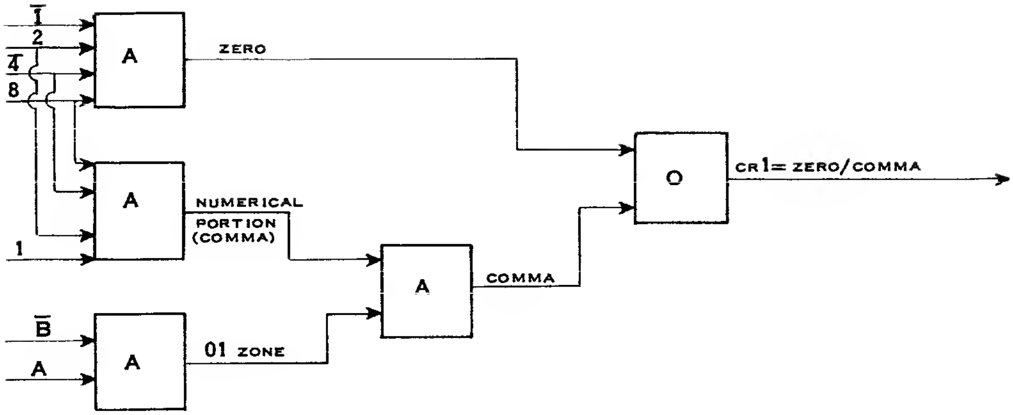
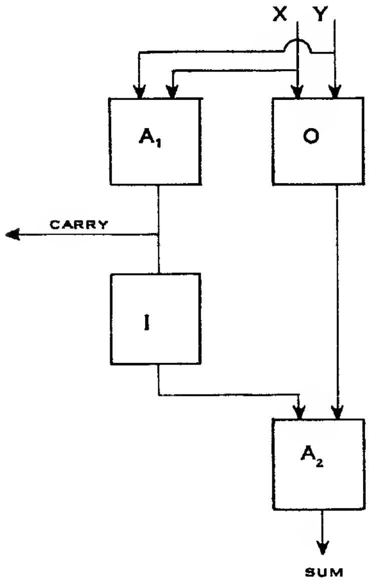


FIGURE 3

As can be seen from the above figure, a zero is recognized in the machine as the presence of an 8 and a 2 bit and the absence of a 4 and a 1 bit. Therefore, the 705 actually identifies four characters as being zero: these are the four characters which have an 8 and a 2 bit only in the numerical portion of their bit structure; 0, $\overset{+}{0}$, $\overset{-}{0}$, and ‡ .

Logical circuitry can be used to produce an electronic unit capable of adding two bits. This unit is called a half adder as no provision is made for a carry-in from another adder. Such an adder, illustrated in Figure 4, adds by the binary number system. On the right-hand side are shown both, the four possible combinations that can exist when two bits are added, and the outputs of the circuits of the half adder corresponding to these four combinations.



CASES	I	II	III	IV
X	0	1	0	1
Y	0	0	1	1
TOTAL	0	1	1	10
OUTPUTS				
O	DOWN	UP	UP	UP
A_1	DOWN	DOWN	DOWN	UP (CARRY)
I	UP	UP	UP	DOWN
A_2	DOWN	UP	UP	DOWN

FIGURE 4

Figure 5 represents a full adder, that is, one capable of adding three binary bits. It differs from the half adder by making provision for a binary carry into the adder.

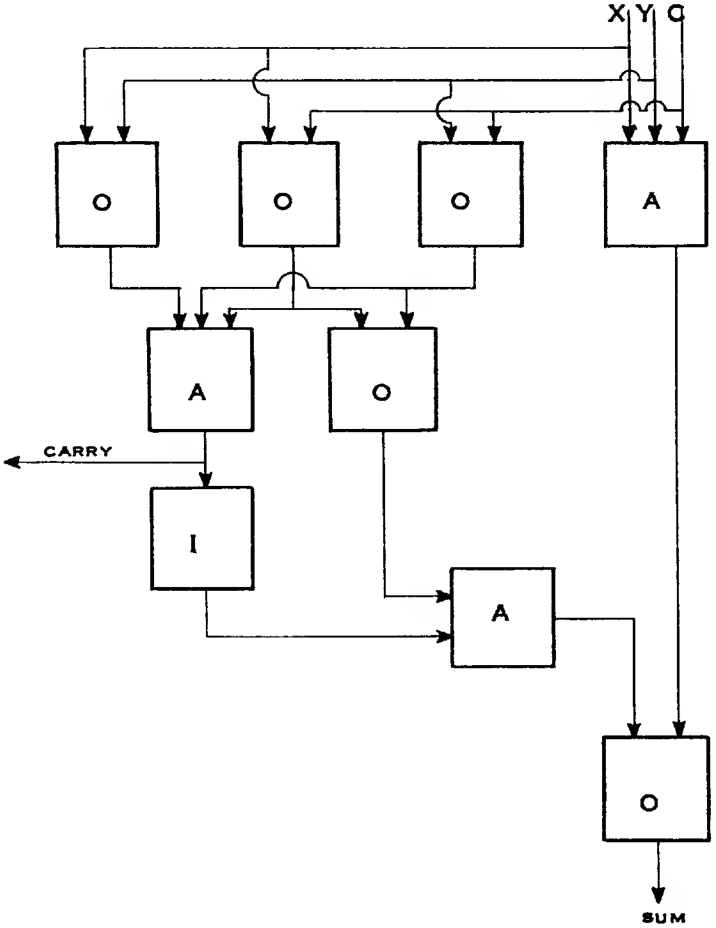


FIGURE 5

As can be seen from the half and full adder, a sum is not actually achieved by addition; but given a set of input conditions, the circuitry arrives at a sum not by addition, but as a result of a series of logical decisions. While the entire unit may be considered an adder, an individual circuit cannot add. All addition is done in binary form, but the resulting digits are automatically changed to binary coded decimal form by means of a decimal correction circuit.

For those interested in the operation of electronic circuits, the IBM manual "Electronics" (Form No. 22-6253) contains, in the section on Electronic Calculators, a description of electrical circuits used to perform logical operations.

TIMING

If the magnetic cores of the 705 are considered as its memory, then an electronic clock is its heart. This clock governs all the operations of the Central Processing Unit (CPU) and operates at a frequency of one megacycle or, stated in another way, it emits a pulse one million times a second. A series or group of these pulses constitute a character cycle, which is the basic unit of operation of the 705. A character cycle normally consists of 17 pulses from the clock or is 17 microseconds (μ sec.) in duration. The cycle can be further broken down into three portions as shown in Figure 6.

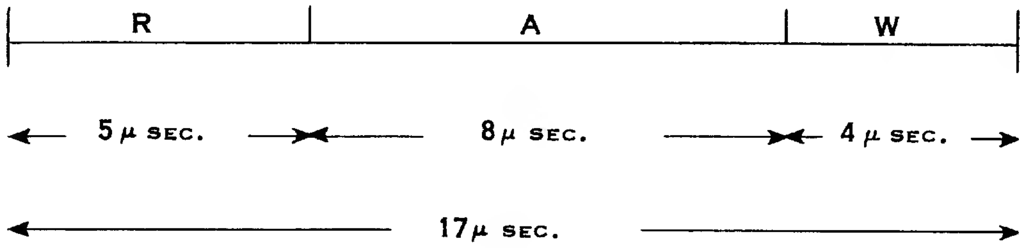


FIGURE 6

The first portion of the character cycle is 5 microseconds in length and is called the R or Read portion. During this part of the character cycle, the character or characters which are to be processed are read from either memory or storage or both. It will be seen from the discussion of 705 memory that characters stored in cores cannot be used there, but must be "read out" and brought into the logical circuits for processing.

During the second part of the character cycle, the A or Arithmetic portion, characters are processed through the adder circuits. Characters always pass through the adder circuitry even though no arithmetic operations are performed, except in the following two cases: in RCV-TMT operations, and in Input/Output operations. When these operations are being performed, the Arithmetic part of the cycles is omitted. Therefore, in these instructions a character cycle is reduced to 9 microseconds, R and W time only. (In the case of Input/Output operations, the 9 microseconds include only CPU time).

The third part of the cycle is the W or Write portion, during which time characters are returned to memory and/or storage.

Timing within a character cycle is shown in Figure 7. The subscript numbers indicate the pulses elapsed since the beginning of that particular portion of the cycle. Note that A_0 and R_5 occur simultaneously; similarly, A_8 and W_0, and W_4 and R_0.

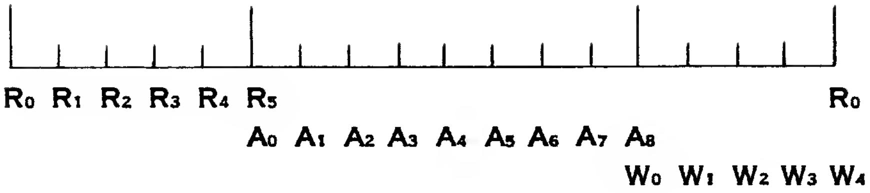


FIGURE 7

This arrangement of pulses is frequently called a "ring", for R_0 time starts again after W_3 time, as shown in Figure 8. The dotted line indicates that, in RCV-TMT and Input/Output operations, part of the cycle is omitted.

The clock may also be considered as an electronic timing gear. Instead of rotating as a timing gear would, however, different triggers are turned on at different times. The clock ring is a series of seventeen triggers arranged in such a fashion that, if trigger 1 is On, trigger 2, and only trigger 2, can be turned on with the next clock pulse. When trigger 2 comes on, trigger 1 is turned off and trigger 3 is set up, so that on the next clock pulse, trigger 3 can be turned on. When trigger 3 comes on, trigger 2 is turned off and trigger 4 is set up, etc. As can be seen, during a regular character cycle, each trigger will be On for one microsecond and only one trigger can be in On status at any given instant. The seventeenth trigger is tied back to the first one so that the ring is complete. These triggers are used to set up various pulses which control all the functions of the machine. The clock may also be thought of as an electronic counter that counts to seventeen and then starts all over again.

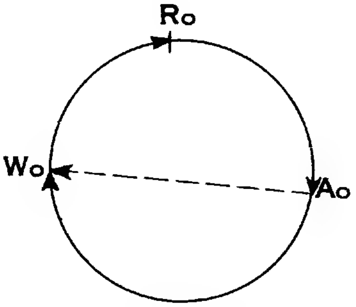


FIGURE 8

MEMORY

The 705 (Model I) has a memory capacity of 20, 000 seven bit characters. To remember these characters the 705 uses magnetic cores, one core being required for each bit. Consequently, 140, 000 cores constitute the 705 memory. These cores are arranged in 35 planes, each containing 4, 000 cores. Such a plane is a rectangle of cores, 50 by 80. The 705 (Model II) is similarly arranged except that each plane consists of a 100 by 80 rectangle of cores.

These 35 planes can be considered as five distinct groups of seven planes each. A group of seven planes will store 4, 000 characters. One plane of a group will contain all the "1" bits of the characters in that group; the next plane will contain all the "2" bits, the next, all the "4" bits, and so on to the "1" plane of the next group. These planes are commonly referred to as "bit" planes. It follows that, in order to get a single character out of the 705 memory, it is necessary to deal with seven planes. Actually, it is impossible to read less than 35 bits, one for each plane, or 5 characters from memory at any one time.

"Reading" a core

Through any one core there are four wires: The X wire. Y wire. sense wire. and inhibit wire as shown in Figure 9. Of these, the X, Y, and sense wires are used in reading a core. The function of the X and Y wires is the selection of the core to be read. Each plane has 50 Y lines and 80 X lines.

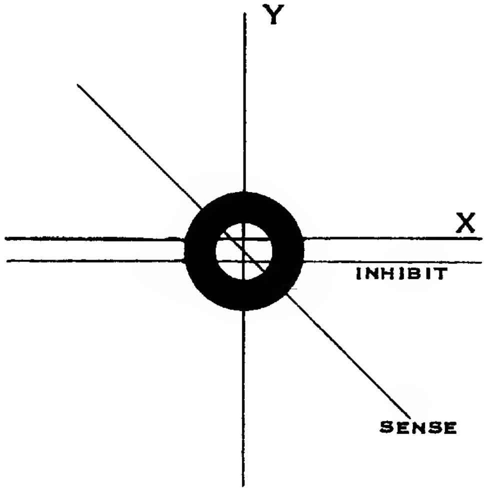


FIGURE 9

Figure 10 shows that if an X line and a Y line in a plane are selected, then only one core will receive a pulse from both the X and Y wires. The other cores along the X and Y lines receive a single X or Y pulse only and will be "half-selected".

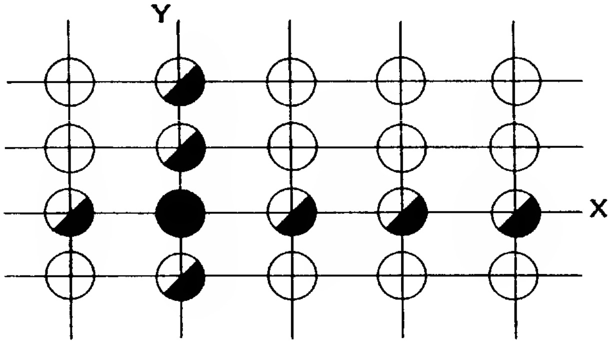


FIGURE 10

Only the core that receives the pulse from both X and Y will be read. This is the result of a magnetic phenomenon known as the square hysteresis loop, which might be best explained by the analogy in Figure 11.

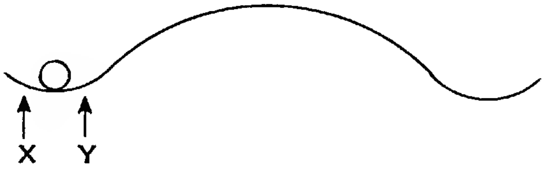


FIGURE 11

Figure 11 represents a hill with a trough on each side and a ball in one of the troughs. Let X and Y be equal forces which may act on the ball, then if an X or a Y force alone is applied, the ball will receive enough impetus to push it about halfway up the hill, and from there it will roll back into the original trough. But if the ball receives the impetus of both an X and a Y force at the same time, then the ball will pass the peak and roll into the trough on the other side of the hill.

The cores react in about the same way to pulses received from the X and/or Y wires: if a core is in either the bit or no-bit condition and receives a single X or Y pulse, there is no change in its condition. If, however, a core in the bit condition receives a read-out pulse from both the X and Y lines, then it will be changed from a bit to a no-bit condition. Figure 12, showing the hysteresis loop, represents graphically the process of reading a core. When a core is changed from a bit to a no-bit condition or "flipped", it goes from the bit position to point A on the graph, then slides to the no-bit position. If the core is in the no-bit condition and is given a read out pulse, it moves from the no-bit position to point A and back again, and thus is not "flipped".

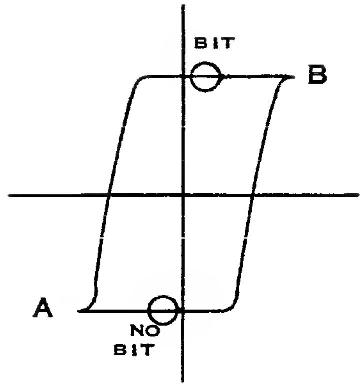


FIGURE 12

A current can be induced in a wire by moving the wire through a magnetic field or by moving the magnetic field while the wire remains stationary. The latter method is used in the 705. When a core is flipped, the direction of its magnetic field is reversed. This reversing of the field is used to determine whether a bit was present in the core, as outlined in the following paragraphs.

It should be noted that in the 705 a core is not magnetized or demagnetized to represent the presence or absence of a bit, but its magnetic field is polarized in either one direction or another as indicated in Figure 13. A round core has no north and south pole as does a bar magnet, but its magnetic field can be either clockwise or counter-clockwise. One direction represents the bit condition and the other represents the no-bit condition.

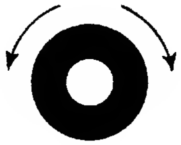


FIGURE 13

The X and Y lines for each of the 35 planes are wired in series. As Figure 14 illustrates, the selection of an X and Y line determines a particular point or coordinate in each of the 35 planes. Therefore, when one X and one Y line are selected, 35 cores are read, one from each plane.

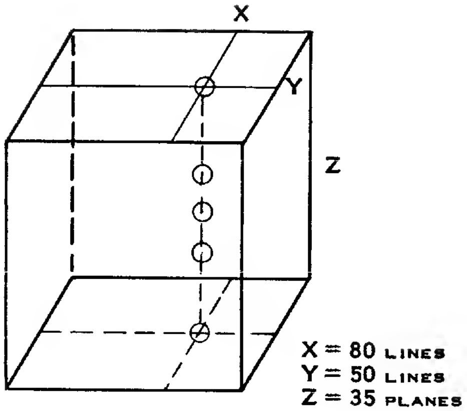


FIGURE 14

In each plane there is a sense wire common to all cores in that plane. The sense wire is wound through each core in the plane in a manner similar to that illustrated in Figure 15. When a core is flipped, there will be an output on the sense wire of that particular plane. If a core is in a no-bit condition when read, some electrical "noise" will be generated in its sense wire. The reason for the unusual winding of the sense wire is to reduce the electrical noise from the half-selected cores. The winding is so designed that, in effect, the noise from the half-selected cores will cancel itself out.

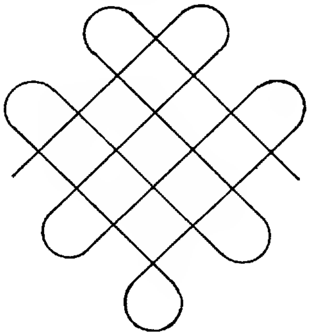


FIGURE 15

Two things are important to note about reading from memory. The first is that all the 35 cores read will be in the no-bit condition at the end of the reading. This is called a "destructive read out". In order to retain the information taken from the cores', it is necessary to write it back at a later time. Secondly, it is impossible to read out of memory less than five characters at a time, since 35 bits are always read. In considering this second point, it can be seen that when five characters are processed in parallel (as they are in I-time and high-speed transmission) it is necessary to have certain address restrictions, which are, on the 705, addresses ending in 4 or 9. In this sense, the 705 has some of the features common to fixed word-length machines.

The five characters read from memory are placed in a set of triggers called the Memory Buffer Register (MBR). When an address is decoded, it is ascertained to which block of five characters it refers and also which character of these five is to be sent from the MBR to Character Register I (CRI).

"Writing" into a core

Writing into a core is effected in much the same way as reading out from the core, and both operations, of course, occur within the same character cycle. The X and Y selection system works in the same way as for reading a core, except that in Figure 12 the magnetic state will change from the no bit position to point B and then slide to the bit position on the curve. Although the same X and Y lines are used as in reading out, the current is sent through them in the direction opposite to the reading pulses in order to cause the cores to pass to bit status. When the X and Y lines are selected, the 705 attempts to write bits into all of the 35 cores previously read, which would produce five seven bit characters.

In order to prevent the forming of seven bit characters, another wire through the cores, the inhibit or Z wire, is used In those planes in which it is not desired to write a bit, the Z or inhibit wire is impulsed at the same time as the X and Y wires. The Z wires then carry a pulse opposite to that received by the X line and so cancel out the X pulses in all relevant planes. The net result is that in those planes that are inhibited, there is, effectively, only a Y pulse and so no core can be fully selected. Inhibit windings, like sense windings, are provided, one for each plane. The inhibit wire is wound in the manner shown in Figure 16.

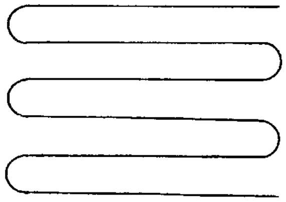


FIGURE 16

In summation:

Figure 9 shows the four wires through each core. The X and Y lines are used so that a coordinate system of selection is possible by determining a coordinate position in each of the planes. The sense windings are used in reading to differentiate between the bit and no-bit condition of a core. The inhibit windings have the function of preventing the writing of all bits in the group of 35 cores selected and are associated, one with each plane, in the same manner as the sense windings. The read-out of memory is destructive and the information must be written back. Repetitive regeneration is not needed with cores, for characters not used do not have to be continually rewritten as necessary in electrostatic storage. Only if a character has been read out, does it have to be written back into core memory.

Core storage, Accumulator and Auxiliary Storage Units (ASU's), operate in essentially the same way as core memory, in that reading and writing are performed in a similar manner. The core planes in core storage are in a rectangle, 16 cores by 32 cores, or a total of 512 cores for each plane. There are seven bit-planes which allow the placement of 512 characters in storage. Only one character is read from, or written into, storage at any one time.

Whenever characters are placed in core memory or storage, the old character is read out, in order to leave a no-bit condition in the cores for the new character. However, the design of the 705 is such that no bits can be written into the cores unless the cores have first been read out. Therefore, the 705 always reads a character whenever that character is to be replaced; e.g., in instructions such as ST and UNL, the memory field in which storage is to be "stored" or "unloaded" is read at the beginning of execution of the instruction. In the LOD instruction, the storage unit selected is similarly read.

CYCLES

Character cycles are classified according to the time in which they occur during an operation. An operation is defined as the period of time during which the 705 reads, interprets, and performs an instruction. The first character cycle of each 705 operation is the Instruction Character Cycle, ICC, or I-Time. During this cycle the five characters of an instruction are read and placed in the various registers, the operation code is recognized, various registers and counters are set, and the characters are written back into memory.

Following I-Time is E-Time or Execution time. The number of cycles in E-Time depends on the type of instruction. In the case of some instructions, the number of Execution Character Cycles varies according to the length of the fields involved, although at least one E cycle is always required in any instruction. The first cycle in E-Time is called ECC1 or Execution Character Cycle which is characterized by certain functions peculiar to it; for example, in arithmetic instructions the 705 must remove the sign from the character addressed in memory; in a SPR instruction, the first character placed in memory is either a blank or a hyphen depending on the setting of the sign trigger of the storage unit. Following ECC1 are as many ECC's or Execution Character Cycles as are necessary to complete the instruction.

Note: At this point it is recommended that the 705 Console Manual be reviewed so that the flow of information through the various console registers and counters will be thoroughly understood.

ADDRESS CONTROLS

Memory

Figure 17 shows the relationship between the counters and registers used to control reading from, or writing into, memory. A register and a counter may be distinguished by the fact that a register can only be set, while a counter may be set or stepped. A counter counts in much the same way that the clock ring, shown in Figure 8, advances. The electronic clock ring might be considered as an unconditional counter, for a clock pulse arrives every microsecond to cause the clock ring to advance or count. The Memory Address Counter I (MAC I), Memory Address Counter II (MAC II), and the Instruction Counter (IC), however, are stepped only when certain conditions are fulfilled; for example, IC is stepped only during I-time if the instruction check stop has not been turned on. These counters actually perform their counting in the binary number system, but are designed to produce a carry to the next position after 9 is reached, that is, they are decimally corrected. In certain instances they can be stepped five positions at one time. MAC I can also be stepped down one position at a time.

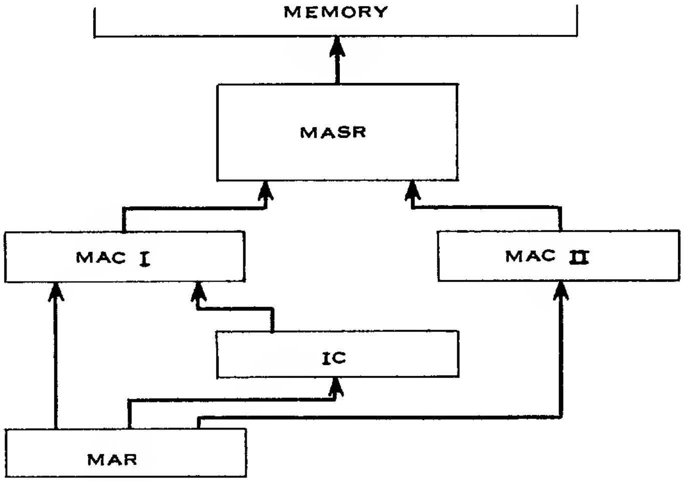


FIGURE 17

Memory Address Select Register (MASR)

The character or characters to be read from, or written into, memory are determined by MASR. MASR can be set from either MAC I or MAC II and is set at R_0 time of every character cycle. The purpose of having two Memory Address Counters, MAC I and MAC II, is to "remember" two different and unrelated memory addresses when RWW or RCV-TMT operations are taking place.

Memory Address Counter I (MAC I)

This counter is stepped at either R_3 or A_6 time, provided there is no machine check stop. The reason for the two different stepping times is that in R-W cycles (i.e. character cycles consisting only of the R and W parts) there is no A_6 time. MAC I can be set to either MAR or IC. It is set to MAR at W_1 of I-time and to IC at W_1 of the last cycle of E-time. (The purpose of MAR is to "remember" the address of the current instruction, and IC is intended to "remember" the location of the next instruction).

Memory Address Counter II (MAC II)

MAC II, when used, is stepped and set at the equivalent times of an execution cycle to those at which MAC I is stepped and set. MAC II is set to MAR during the execution of RWW or RCV instructions only.

Instruction Counter (IC)

Stepping occurs at A_6 of I-time provided there is no instruction check stop. IC is set to MAR at A_5 of ECC1 of an executed transfer instruction. It can also be set to 00004, when thc 705 is in Manual status, by means of the Reset or Clear Memory keys.

Memory Address Register (MAR)

This register is set at A_5 of I-time to the address of the current instruction. It can also be set directly from the console by a manual instruction when the 705 is in Manual status.

Storage

Figure 18 shows the relationship between the various counters used to control reading out of, or writing into, the accumulator and the ASU's.

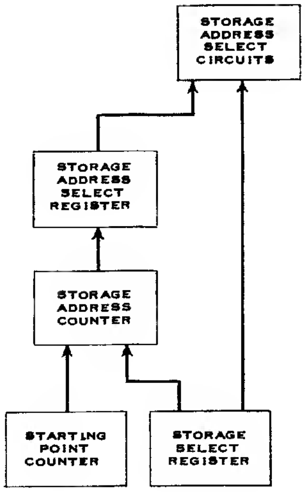


FIGURE 18

Storage Select Register (SSR)

This register performs for the storage units the same function as MAR does for memory. The SSR is set from either the MBR, or from the console by a manual instruction when the 705 is in Manual status. The SSR is set at A_0 time of each instruction character cycle. One line from SSR goes directly to the storage address select circuits, which performs only one function: to determine whether auxiliary storage or the accumulator is selected.

Storage consists of 512 positions. The accumulator is in positions 256 through 511, and the ASU's are in positions 000 through 255. The line from the SSR to the storage address select circuits indicates whether a position equal to or above 256, or a position lower than 256, is addressed. This line, which indicates whether SSR equals zero or not, is also used to cause a change in some of the instructions. For example, execution of a WR (00) instruction is ended by the sensing of a Group Mark, while WR (01) is terminated after writing the contents of memory position 19998 (in a 705 Model I).

The second line leads to the Storage Address Counter (SAC) and sets that counter, if $SSR \neq 0$, to the fixed storage address indicated by the ASU number shown in SSR. This fixed storage address is 000 for ASU 01, 016 for ASU 02, etc. up to storage address 224 for ASU 15.

Starting Point Counter (SPC)

The only function of SPC is to determine the beginning of the accumulator. This counter can count from 000 through 255. It counts in straight binary as distinguished from other counters, such as MAC I, which count in binary but are decimally corrected. SPC can be stepped either plus 1, minus 1, or plus 128 at a time; it is stepped plus 1 or minus 1 at A_6 time, and plus 128 at W_1 time. When the counter passes 255, it starts off again at zero. Stepping only can occur in five instructions: SHR, LNG, RND, MPY, and DIV.

Storage Address Counter (SAC)

SAC is set to either SPC, if SSR is equal to zero, or to a fixed storage address (000, 016, 032, 048, ...224) depending on the ASU indicated in SSR. Like SPC, it is a counter capable of counting from 000 to 255 in the binary system. It is set at W_1 time, and stepped at A_6 or W_1 time in the same way as SPC. SAC performs for storage the same function as MAC I carries out for memory. It is used to set SASR which, in turn, determines the character to be read from, or written into, the accumulator or an ASU.

Storage Address Select Register (SASR)

This register is set to SAC at R_0 time, at the same point in the cycle that MASR is set to either MAC I or MAC II. As mentioned above, it serves the same function for storage that MASR serves for memory: selection of the character to be read from, or written into, storage.

In summation:

In considering the overall flow of information through these counters and registers for storage address control, SSR determines two things:

1. Whether the storage address select circuits will be set to either the upper or lower half of storage.

2. Whether SAC will be set to SPC or to the fixed storage address corresponding to the ASU indicated in SSR.

The "starting point" is simply an address in storage. When any of the ASU's are used, this is a fixed address; but when the accumulator is selected, the starting point may be any address of the upper 256 positions of storage. The SPC has as its only function to "remember" the location of the starting point in the accumulator.

GENERAL FLOW OF INFORMATION

Figure 19 is a schematic of the overall flow of information through the 705. At this point a general description of the various components is given. When each component is first used in an instruction, all detail herein omitted will be included in the explanatory material on the relevant instruction.

All information read from core memory is normally placed in MBR. From there the Memory-Out Switch determines which of the five characters will be placed in CR1. All information read from core storage is placed in the Storage Buffer Register (SBR). From there it is moved to CR2. Since only one character at a time is read from storage, as distinguished from the five characters read from memory, no Storage-Out Switch is needed. If the Result Register is not routed to memory, then the original characters read from memory are returned there through the Memory-In Switch during the W portion of the character cycle. The same is also true of the Storage-In Switch: if the contents of the Result Register are not sent to storage, then the original character is returned to storage.

All character recognition takes place in CR1 and CR2. Information comes into CR1 from memory only. From CR1 information can go to the Input/Output (I/O) write buses, to the digit adder through the multiply circuits, or to the digit adder through the true complementer (TC) and to the zone adder. Information can be moved into CR2 from either the I/O read buses or from storage. From CR2 information can go either to the digit adder through the true complementer and to the zone adder, or to the multiply register. After a character leaves CR1 or CR2, it is split up into its three component parts: the numerical portion, the zone portion and the C-bit.

The numerical portion of a character leaving CR1 normally passes through the multiply circuitry which is always set to multiply by one unless the instruction is MPY. In the ADM instruction, the numerical portion may be routed to the digit adder through the true complementer. The numerical portion of a character leaving CR2 is moved to the digit adder through the true complementer. If the true complementer is signaled to complement, it produces the 9's complement of the number sent to it. If it is not signaled to complement, the information passes through it in unchanged form. In the MPY instruction, the numerical portion of the character leaving CR2 goes to the multiply register where it acts as the multiplier.

In the digit adder, numbers are added in binary form. However, the circuitry is so arranged that the numbers are always decimally corrected unless this correction is suppressed. In some cases, such as LOD and UNL instructions, it is necessary to pass information through the digit adder without decimally correcting it. For example, a $ sign has a numerical portion of eleven, and in order to prevent a decimal correction, the adder carry is suppressed. Thus, information may pass through the digit adder unchanged. The decimal carry-in line is in an Up condition whenever there is a carry from one digit to the next.

The 705 cannot subtract directly, but adds the 10's complement of one of the numbers to the other number, ignoring the carry as illustrated below:

Direct Subtraction	The Same Subtraction by Complementing
68	68
-59	41
09	c09
	a
	r
	r
	y

The true complementer only produces the 9's complement. Therefore, when a number is to be complemented, a decimal carry-in is forced during the first character cycle, which causes an extra 1 to be added to the first digit complemented. This will produce the 10's complement of any required number, as the 9's complement of a given number plus 1 equals the 10's complement of that number.

In most cases, the zone adder does not add zones but simply passes information through in unchanged form. In the ADM instruction, the zones may be added, and in the CMP instruction the characters are compared by subtracting them, including their zones. Thus, it is necessary to be able to complement zones and to produce a carry-in.

The Digit Adder-Out Switch is used to determine whether the output of the digit adder is allowed to reach the Result Register. Similar functions are performed by the Zone Adder-Out Switch, the CR1 C-Bit Switch, and the CR2 C-Bit Switch.

The Character Emitter (CE) is used to emit certain characters when necessary. It can emit plus, minus or zero zones, and the number zero. The C-Bit Generator is used in instructions where, for various reasons, the original C-bit of the character or characters cannot be used. This occurs in general when new characters are produced, as in arithmetic instructions.

In the Result Register, the three parts of the characters processed are assembled again:

The numerical part	from the Digit Adder or Character Emitter.
The zone part	from the Zone Adder or Character Emitter.
The C-bit	from the CR1 or CR2 C-Bit Out Switch or from the C-Bit Generator.

From the Result Register, the character is routed either to memory or to storage during the W portion of the character cycle.

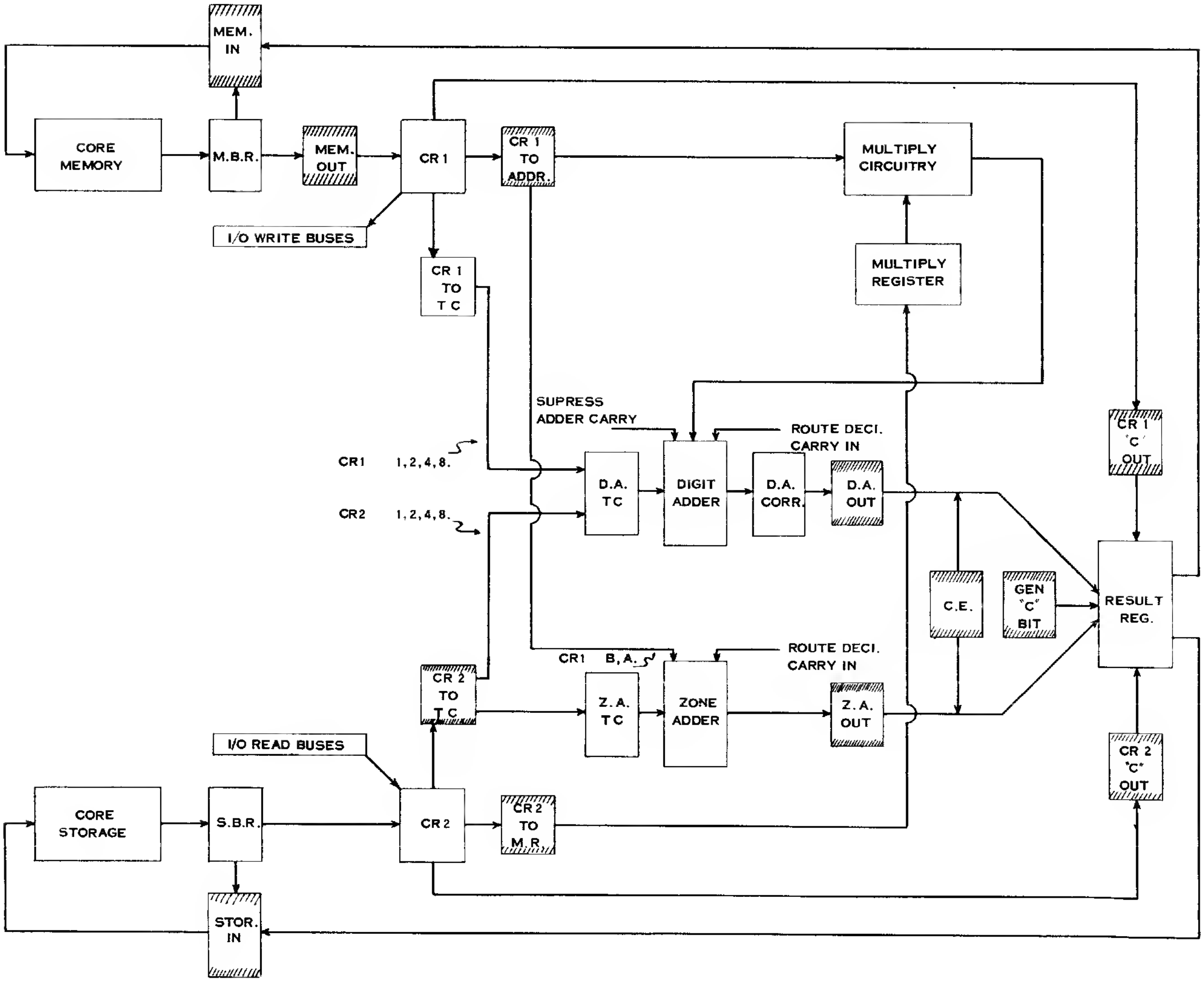


FIGURE 19

INSTRUCTIONS: INSTRUCTION TIME

In a stored program computer, such as the 705, the computer has no way of distinguishing between instructions and data, except by the time, either I-or E-time, when it reads the information:

If the 705 is in I-time, then the information is an instruction.
If it is in E-time, then the information is data to be operated on.

Even in I-time the information read must be operated on in some way; therefore, I-time may be considered as a separate and set course of action (or instruction) inserted before every E-time. Seen in this light, I-time serves as an "instruction" to be carried out.

When the 705 reaches the limiting factor in the execution of an instruction, it brings a line called "End E-Time" into an Up condition, which causes the next cycle to be an instruction character cycle. In that last execution character cycle, MAC I is set to IC at W_2 time.

Figure 20 represents a time table of the various actions carried out during I-time. At R_0 time the following registers are reset in order to prepare them for the new information to be received: MAR, SSR, Operation Register and Operation Decoder. Simultaneously, MASR is set to MAC I in order that the five characters of the instruction can be read from memory. At $R_{3.5}$ time, the Memory Buffer Register is set to the five characters read from memory (the instruction). The Operation Register, SSR and CR1 are set to the new information at A_0, and MAR is set to MBR at A_5.

At A_6, there is a conditional operation: IC is stepped plus 5 if no instruction check stop is indicated. Depending on whether SSR is equal to zero, or not, SAC is set to either SPC or SSR at W_1 time. Simultaneously, MAC I is set to MAR.

At W_2 time the instruction is written back into memory, and the E-time trigger and ECC1 trigger, indicating Execution time and ECC1, respectively, are turned on. Triggers used to indicate different types of cycles are, in general, turned on a W_2 time in order to allow them to "settle down" before the beginning of the next cycle. Thus, it may be said that the machine cycle runs from W_2 time to W_2 time, while a memory cycle runs from R_0 time to R_0 time.

MBR is cleared at W_3 time to be ready to receive data in E-time. At W_4 time, which is also R_0 time of ECC1, the memory and storage address select registers are set to MAC I and SAC, respectively, so that the proper character may be read from memory or storage or both. SASR and MASR are so set each I-time, although in fact characters may not be read from either memory or storage, for instance, in the case of the NOP instruction.

I—TIME

R_0	set MASR to MAC I reset MAR, SSR, Operation Register
R_1	
R_2	
R_3	
$R_{3.5}$	set MBR
R_4	
A_0	set CR_1, Operation Register, SSR
A_1	
A_2	
A_3	
A_4	
A_5	set MAR to MBR
A_6	step IC + 5 (if no instruction check)
A_7	
W_0	
W_1	set SAC to SPC or SSR ; set MAC I to MAR
W_2	write instruction back into memory; turn on ECC I trigger turn on E–time trigger
W_3	reset MBR
R_0	set MASR to MAC I ; set SASR to SAC

note: W_2 of last E-cycle (end E–time) sets MAC I to IC.

figure 20

INSTRUCTIONS: EXECUTION TIME

NOTE:
To the left of each figure, where possible, are shown the times within each cycle at which various instruction phases happen or subsequent effects are caused. The ① circle at the beginning of each diagram indicates the point at which E-time starts.

NOP
As the NOP instruction (Figure 21) is the simplest instruction the 705 performs, it is shown first. Only one function is performed by a NOP instruction: in the first cycle of E-time or ECC1, E-time is ended.

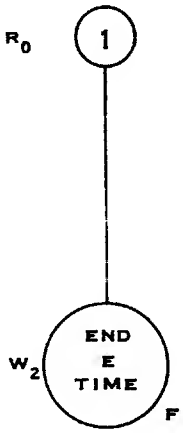


FIGURE 21

HLT
The HLT instruction (Figure 22) performs the same function as the NOP instruction with one addition: at W_1 time the Stop trigger is turned on to prevent the next character cycle from starting.

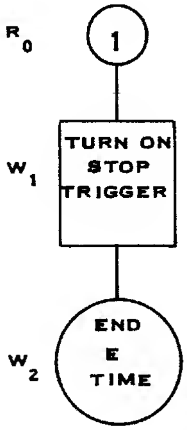


FIGURE 22

TR

The unconditional TR instruction (Figure 23) resets IC, sets IC to MAR, and ends E-time at the end of ECC1.

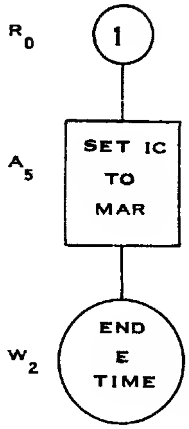


FIGURE 23

TRP/TRZ/TRH/TRE

In these four conditional transfer instructions (Figure 24), the transfer is executed only if the proper trigger is On. In each of these four instructions, the trigger interrogated is not turned off by the interrogation. There are two sets of plus and zero triggers: one set serves the accumulator and the other set serves auxiliary storage. As there is only one set of zero and sign indicators for all ASU's, the status of these indicators is dependent upon the last instruction executed in any auxiliary storage unit. Note, however, that not all instructions referring to a storage unit are capable of activating these indicators. There is only one high and one equal trigger for the entire machine.

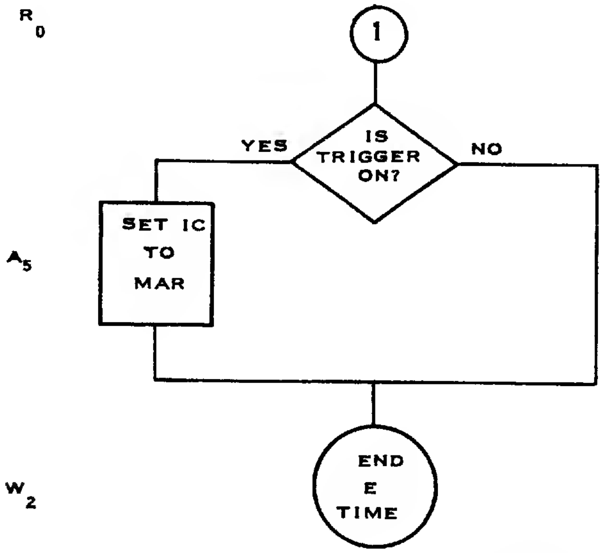


FIGURE 24

TRA

The TRA instruction (Figure 25) is also a conditional transfer, but is distinguished from the conditional transfers perviously mentioned in that the transfer causes the trigger to be turned off.

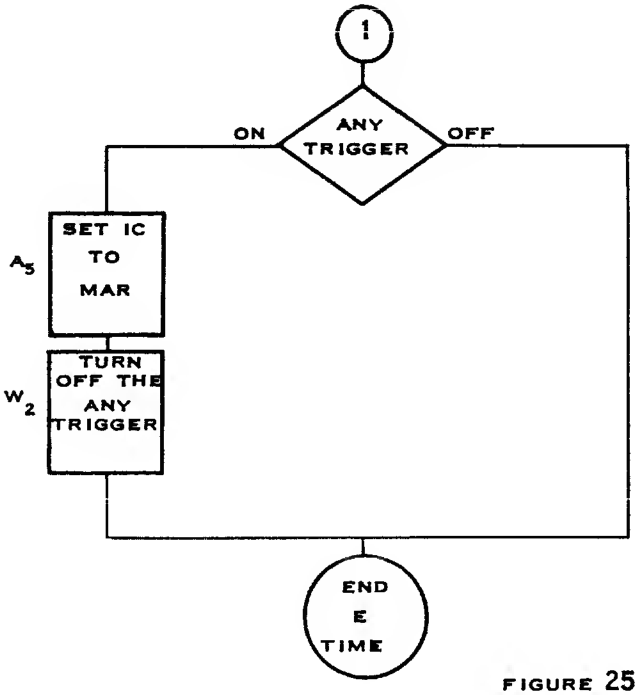


FIGURE 25

TRS

With the TRS instruction (Figure 26), it is necessary that several decisions be made during E-time. The first decision concerns the status of the SSR: is it equal to zero or not. The purpose of this decision is to distinguish between a TRS instruction and a Transfer Ready instruction which is used only with the 777 Tape Record Coordinator (TRC). This is an example of how ASU zoning may change the nature of an instruction although no storage units are actually used. If there is no TRC in the system, then a transfer will not be made and the 705 will progress to the next sequential instruction. If SSR is equal to zero, then certain decisions, shown in the diagram, are made.

While these decisions are shown separately, many of them are actually made simultaneously by means of And and Or gates, according to the condition of the check indicators, alteration switches, and EOF indicators.

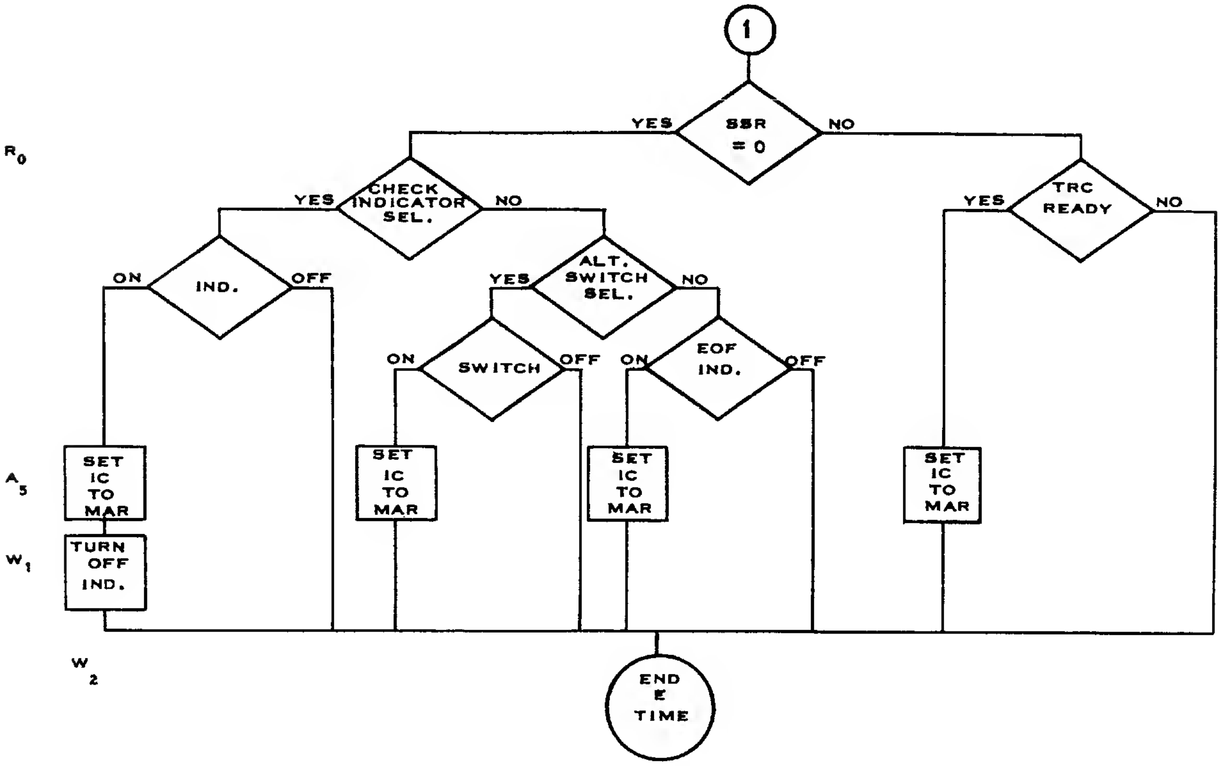


FIGURE 26

RCV

The RCV instruction (Figure 27) performs only one function: it sets MAC II to MAR. No check is made to determine if MAC II is set to an address with the units position ending in 4 or 9, as, at this point in time, the 705 does not know whether high speed or serial transmission will follow.

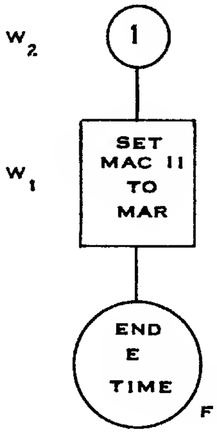


FIGURE 27

RWW

The RWW instruction (Figure 28) performs three functions:

1. It generates a Prepare to Read signal used in the Tape Control Unit to remember that the input tape previously selected shall continue in select status.
2. MAC II is set to the address contained in MAR.
3. A trigger is turned on which indicates to the following RD or WR instruction that a simultaneous read-while-write operation is to take place.

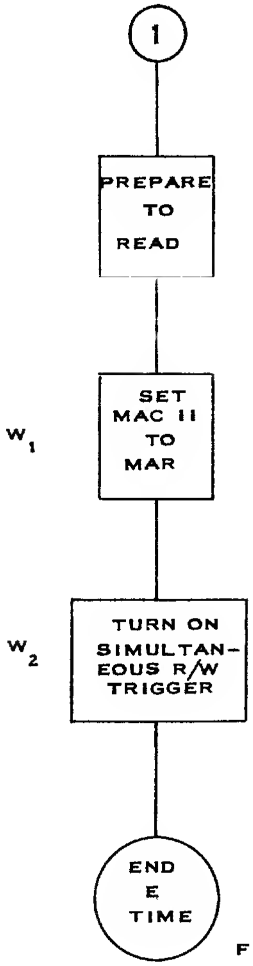


FIGURE 28

SEL

The SEL instruction (Figure 29) performs only one function: it sets the Select Register to MAR. Although only one cycle is required to set the Select Register, two cycles of E-time are used in order to allow the select lines to settle down before the execution of the next instruction. This is the only purpose of the second cycle of E-time.

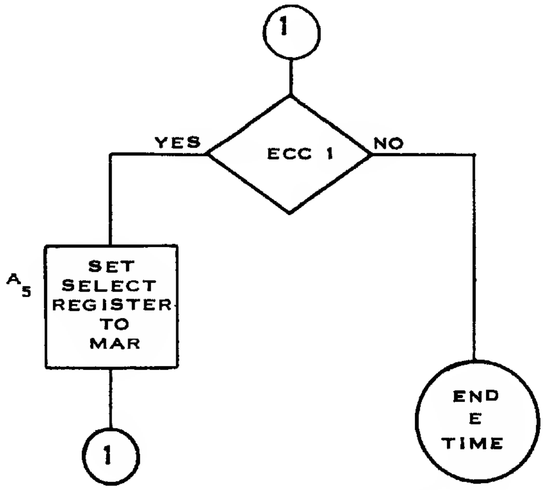


FIGURE 29

LOD

The purpose of the LOD instruction (Figure 30) is to move characters from memory to storage. Operations in the upper left corner of Figure 30 are performed unconditionally during each character cycle:

Memory is read, placed in CR1, and sent to the adder.
Storage is read in order to erase the character that was present in storage.
The Suppress Adder Carry is used in order to prevent a decimal correction of characters having a numerical portion of ten or larger. Thus, special characters are allowed to pass through the digit adder unchanged.

During ECC1, two functions are performed in addition to those occurring in the other Execution character cycles. The Digit Zero Trigger (DZT) is turned on, and the storage sign is set to plus. As previously indicated, there are two zero triggers and two plus triggers in the 705, and the setting of SSR, whether equal or unequal to zero, will determine which pair of triggers is turned on.

After ECC1, characters will be "loaded" until the Storage Mark Trigger (SMT) is turned on by the reading of a storage mark. As long as the SMT is Off, the C-bit from CR1 and the zones from the zone adder are sent to the Result Register.

CR1 recognizes, among other things, a blank, hyphen, or ampersand, all of which have no numerical bits. If the digit adder receives no bits as input, it will produce an 8 and a 2 bit as output to represent a zero. If one of these characters is sensed, the output of the digit adder is suppressed. Therefore, the Digit Adder-Out Switch is not turned on if CR1 is equal to a blank, hyphen, or ampersand, and, in this manner, the proper bit structure of these characters is retained. At A_6, both MAC I and SAC are stepped in order to load the next character during the following character cycle.*

If CR1 is odd, the machine check trigger is turned on at W_1 time. The condition, however, is recognized as soon as the character is placed in CR1. The recognition of an error condition will prevent MAC I, MAC II, SAC, or SPC from stepping during the particular character cycle in which the error condition occurs if the 0901 switch is on Automatic; although the machine check trigger is not turned on until W_1 time. It is the recognition of the error condition which prevents the stepping of these counters, not the status of the machine check trigger. This statement is quite generally true and applies to all subsequent instructions. In order to avoid repetition, it will not be mentioned again.

If the 0901 switch is set to Automatic, the machine will stop at the end of the current cycle. Note that the characters read from storage to be erased are not checked, since the only purpose in reading them is to erase them.

The contents of the Result Register are sent to storage, and the character which originally was in memory is returned there.

The DZT is turned off whenever a non-zero character is found. The characters are loaded until a storage mark is reached, at which time CR2 is also checked for a redundancy. The reason for the CR2 code check lies in the way in which CR2 recognizes a storage mark: a storage mark is recognized by the absence of zone and numerical bits. If a C-bit only were present, then this redundant character would still be recognized as a storage mark. Consequently, during the cycle in which the SMT is turned on, a check is made of CR2 to determine that a valid storage mark was read. A character cycle must be taken to read the storage mark on a LOD instruction, as only in this way will the 705 know when to end E-time.

*In this instruction, and quite generally, the stepping of MAC I or II, SAC, and the other counters during the middle of the cycle does not affect the routing of characters being processed during that cycle, as MASR and SASR (the registers which control character reading and writing) are set to MAC I or II and SAC only at the beginning of a character cycle.

LOD

UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CR1 TO ADDER
SUPPRESS ADDER CARRY

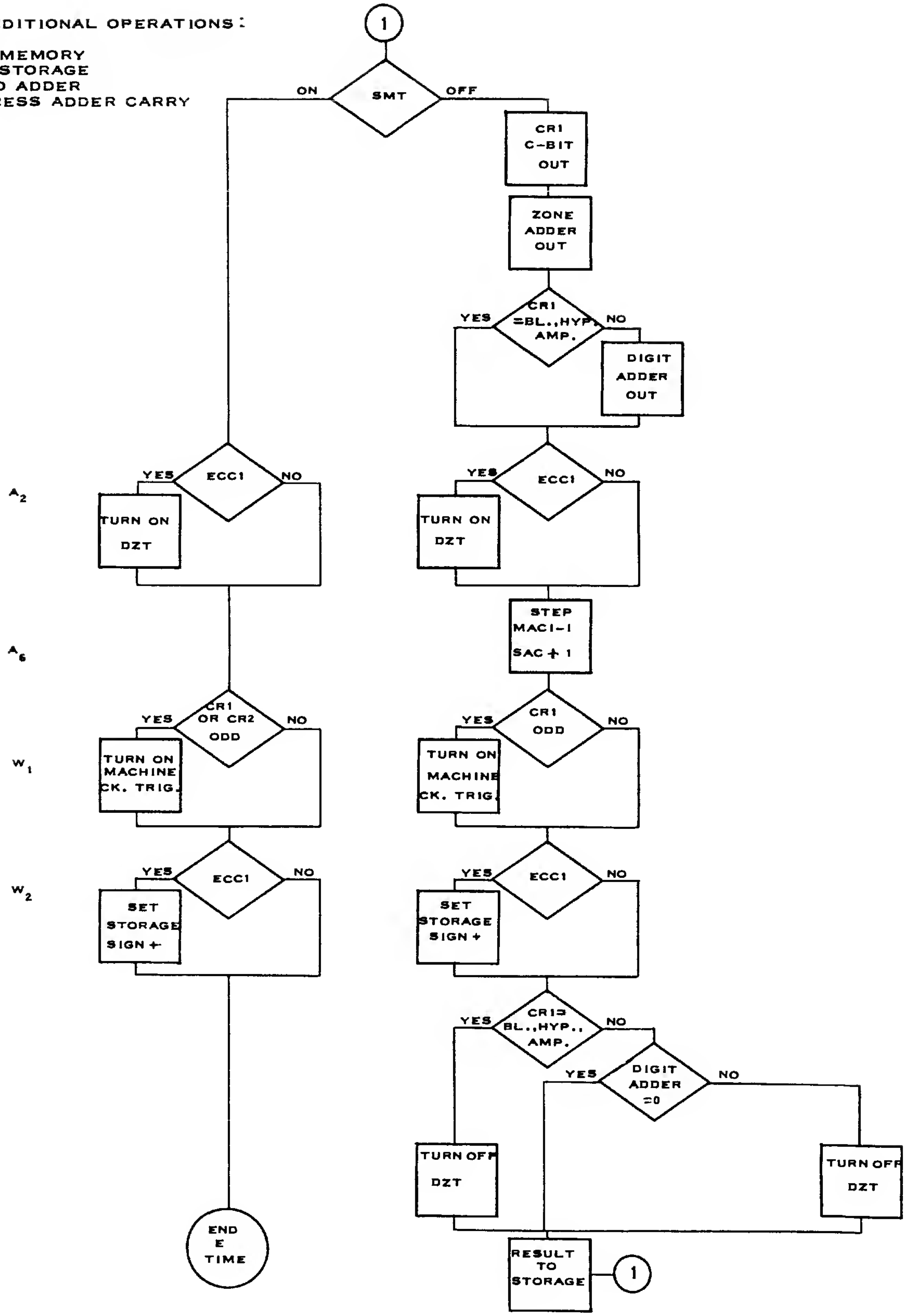


FIGURE 30

UNL

The UNL instruction (Figure 31) is similar to the LOD instruction. Neither the storage sign nor the DZT is, however, disturbed. CR2 is sent to the True Complementer, as only in this way can characters from CR2 reach the adder circuits. As the TC is not signaled to complement, the characters will not change in passing through it. The contents of the Result Register are sent to memory, therefore the original characters will be returned to storage by the Storage-In-Switch.

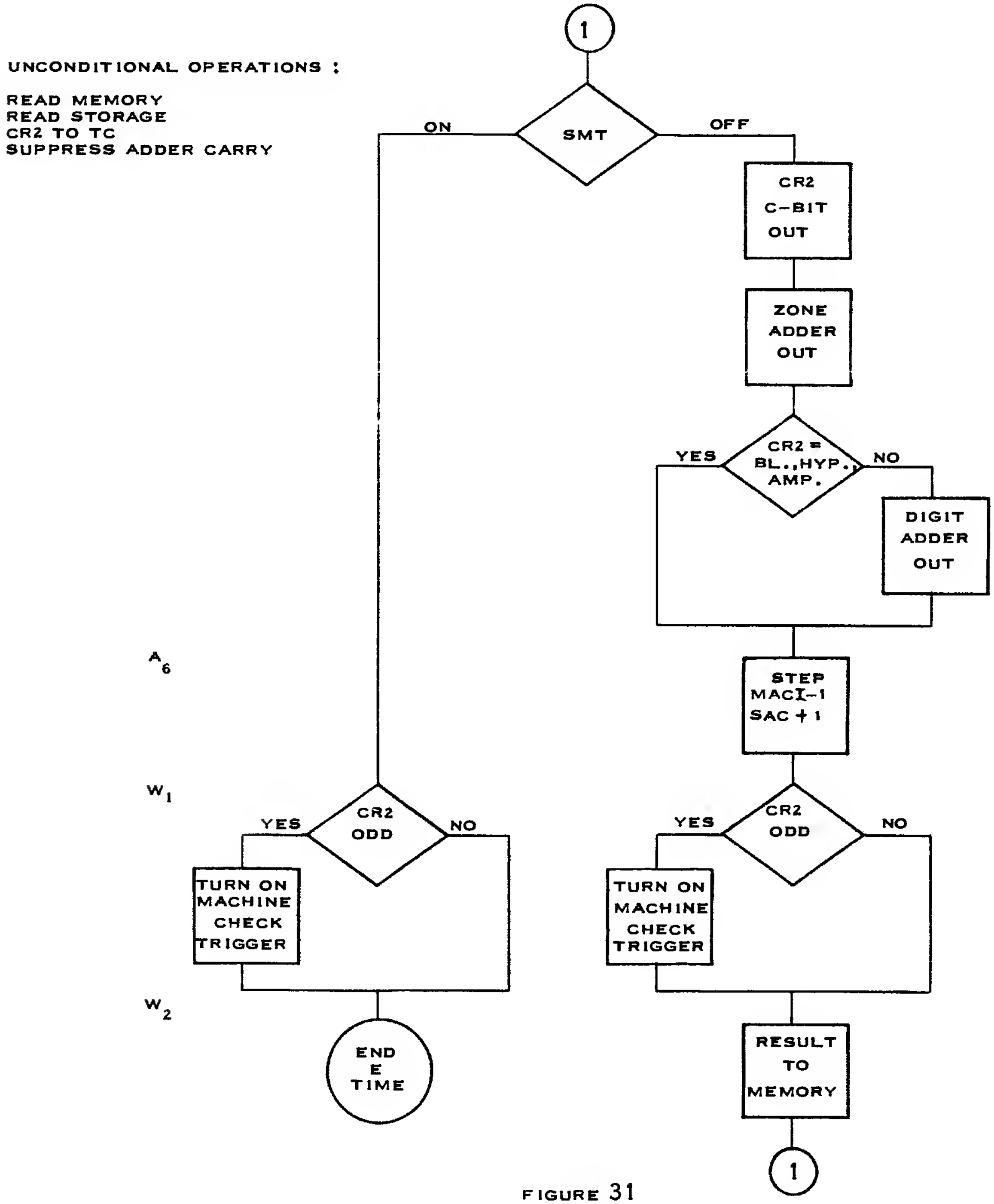


FIGURE 31

ST

The ST instruction (Figure 32), unlike LOD and UNL, is designed to work with numerical characters only. In ECC1 the sign of storage is placed over the first character to be stored. Since either an odd or an even number of bits may be added to the character (the plus zone consists of B and A bits, and the minus zone of a B bit only), the C-Bit Generator is signaled to operate. This generator may be thought of as a redundancy check in reverse: if an odd number of bits (exclusive of the C-bit) is present in the Result Register, the C-Bit Generator will supply a C-bit; if there is an even number of bits, it will not do so.

After ECC1, as long as the storage mark trigger is not On, the C-bit, if any, of the character being stored is used. The Digit Adder Out-Switch is activated as long as the SMT is Off, and thus, if a blank, hyphen, or ampersand is stored, it will acquire a numerical portion of 8 and 2. If a zoned character is stored, the zones are removed since the zone adder is not signaled to operate; and, because the original C-bit of the character is used in positions other than the first, redundancies may be created in those positions. A machine check, however, will not occur during this instruction because the character in CR2 will be correct. The character will also be returned to storage correctly, as the Storage-In Switch returns to storage the same character as was removed, unless, as previously noted, the contents of the Result Register are sent to storage.

When the storage mark is read and recognized, the character read from memory at that time will be the one to the left of the field stored. It is examined to determine whether or not it is numerical. If numerical, a plus-zone is emitted to sign that character. The original C-bit, if any, is used in all cases, as a plus-zone has two bits which would not change the number of bits in the character from even to odd. If CR1 is not numerical, the zone adder is signaled to operate, and thus to return the original character to memory. As in other instructions, the Digit Adder-Out Switch is turned on depending on the absence of a blank, hyphen, or ampersand in CR1 at this time.

It can be seen from the diagram that SAC is stepped irrespective of the condition of SMT. When SMT is On, the stepping of SAC is non-functional, but will not interfere with machine operation. Machinewise, however, it is often easier to step a counter unconditionally than conditionally, as this reduces the number of decisions the machine must make.

ST

UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
SUPPRESS ADDER CARRY
RESULT TO MEMORY

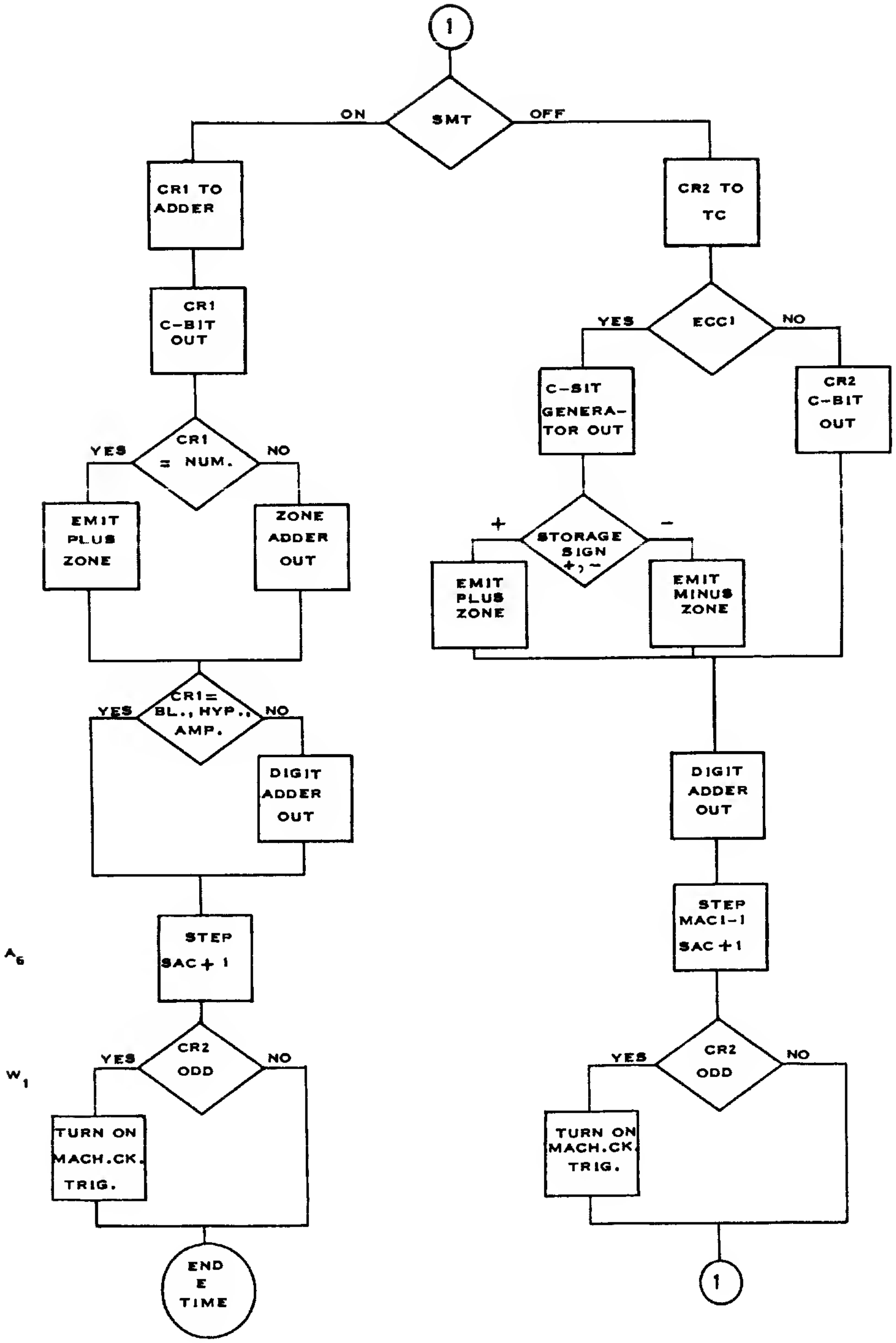


FIGURE 32

SGN

The SGN instruction (Figures 33, 34, 35, and 36) has three types of character cycles, called type cycles. The different type cycles are used to distinguish between functions performed at different times during the execution of the instruction.

Type Cycle I has as its function to place the plus or minus sign in storage, set the appropriate storage sign trigger, turn off the DZT and step SAC so that a storage mark can be entered during Type Cycle II.

At the end of Type Cycle I, Type Cycle II starts unconditionally. In this cycle a storage mark is entered to the left of the plus or minus sign. This is accomplished by reading storage and sending the contents of the Result Register to storage without introducing any information into the Result Register. As has been previously noted, if the contents of the Result Register were not sent to storage, the character originally read from storage would be returned.

In Type Cycle III, the character is read from memory again and the zones removed. The C-bit Generator is signaled to operate, and a decision is made either to route out the digit adder or to emit a blank if the character in CR1 is a blank, hyphen or ampersand. Therefore, if any of these three characters is addressed by a SGN instruction, it is replaced in memory by a blank.

OVERALL FLOW

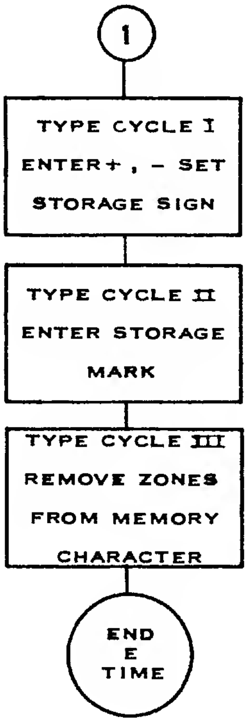


FIGURE 33

SGN

TYPE CYCLE I
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
C-BIT GENERATOR OUT
RESULT TO STORAGE
TURN OFF DZT

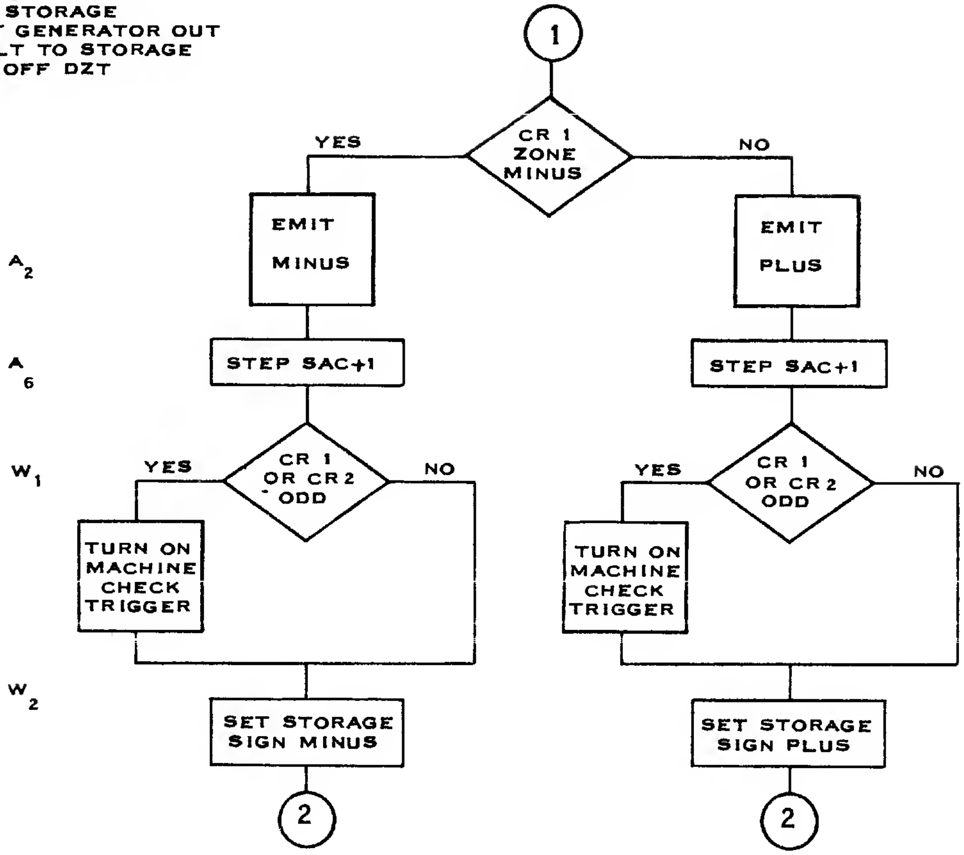


FIGURE 34

SGN

TYPE CYCLE II
UNCONDITIONAL OPERATIONS:

READ STORAGE
RESULT TO STORAGE

FIGURE 35

SGN

TYPE CYCLE III
UNCONDITIONAL OPERATIONS:

READ MEMORY
CRI TO ADDER
SUPPRESS ADDER CARRY
C-BIT GENERATOR OUT
RESULT TO MEMORY

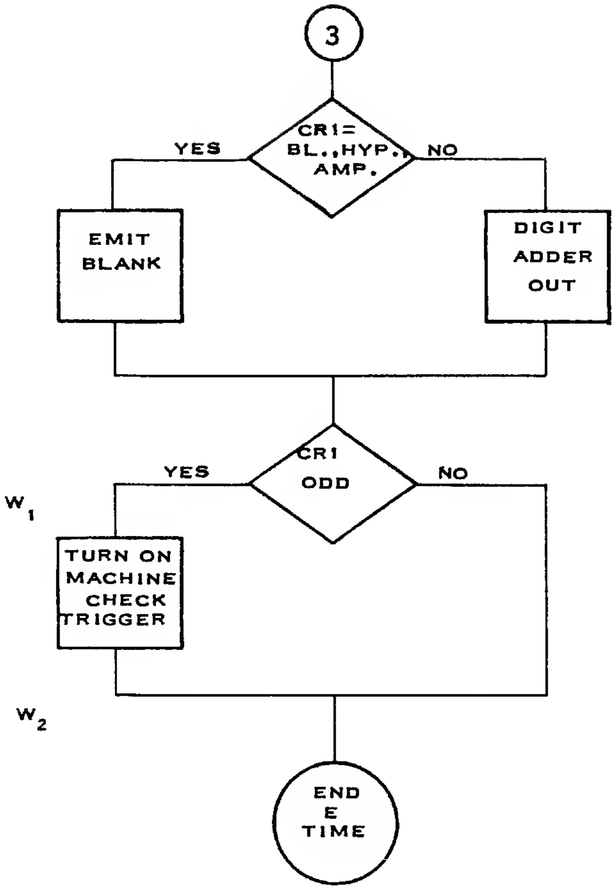


FIGURE 36

NTR

During Type Cycle I of the NTR instruction (Figures 37, 38, 39, and 40), SAC is stepped up until a storage mark is reached. (During I-time SAC is always set to either SPC or SSR depending on whether or not SSR is equal to zero). When a storage mark is sensed, SAC is stepped back by one so that, during Type Cycle II, the character can be examined to determine if it is a zero. Another trigger, the Auxiliary Trigger (AT), is used in Type Cycle I. It is a miscellaneous trigger which is used whenever needed. In Type Cycle I of a NTR instruction, it is used to determine if the field in storage has a length of one. As the diagram shows, AT is turned on during ECC1, and, if in the next cycle both SMT and AT are On, then E-time is ended immediately. If SMT is Off, indicating a field with a length other than one position, the AT is turned off and SAC is stepped until the storage mark is reached, at which time SAC is stepped down one position.

In Type Cycle II, SAC is at the left end of the storage field. This character is examined to determine if it is a zero. If so, then Type Cycle III follows, during

OVERALL FLOW

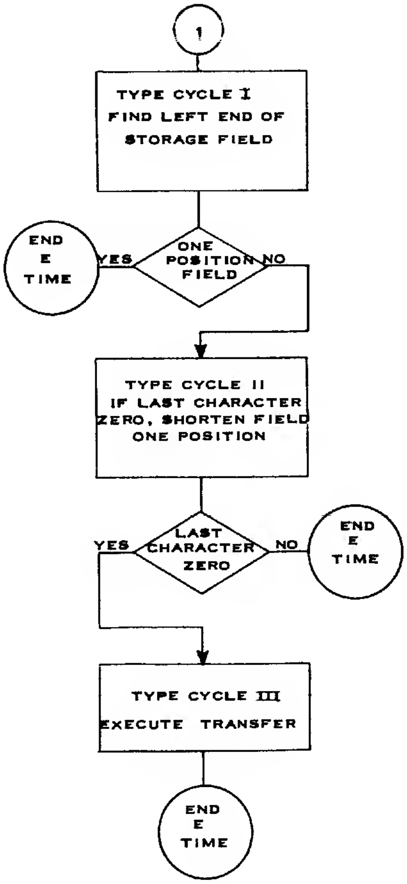


FIGURE 37

which the transfer is executed. If CR2 does not contain a zero, then E-time is ended at the end of Type Cycle II.

If a zero is recognized in CR2 during Type Cycle II, a storage mark is entered in place of the zero by sending the contents of the Result Register to storage without routing any character to the Result Register.

It may be seen from the diagrams that a transfer may be executed if storage is set to zero, i.e., if the SMT is On at the start of execution of the NTR instruction, provided the character in CR2 is a zero in the Type Cycle II. (See also 702/705 Customer Assistance Bulletin 3).

NTR

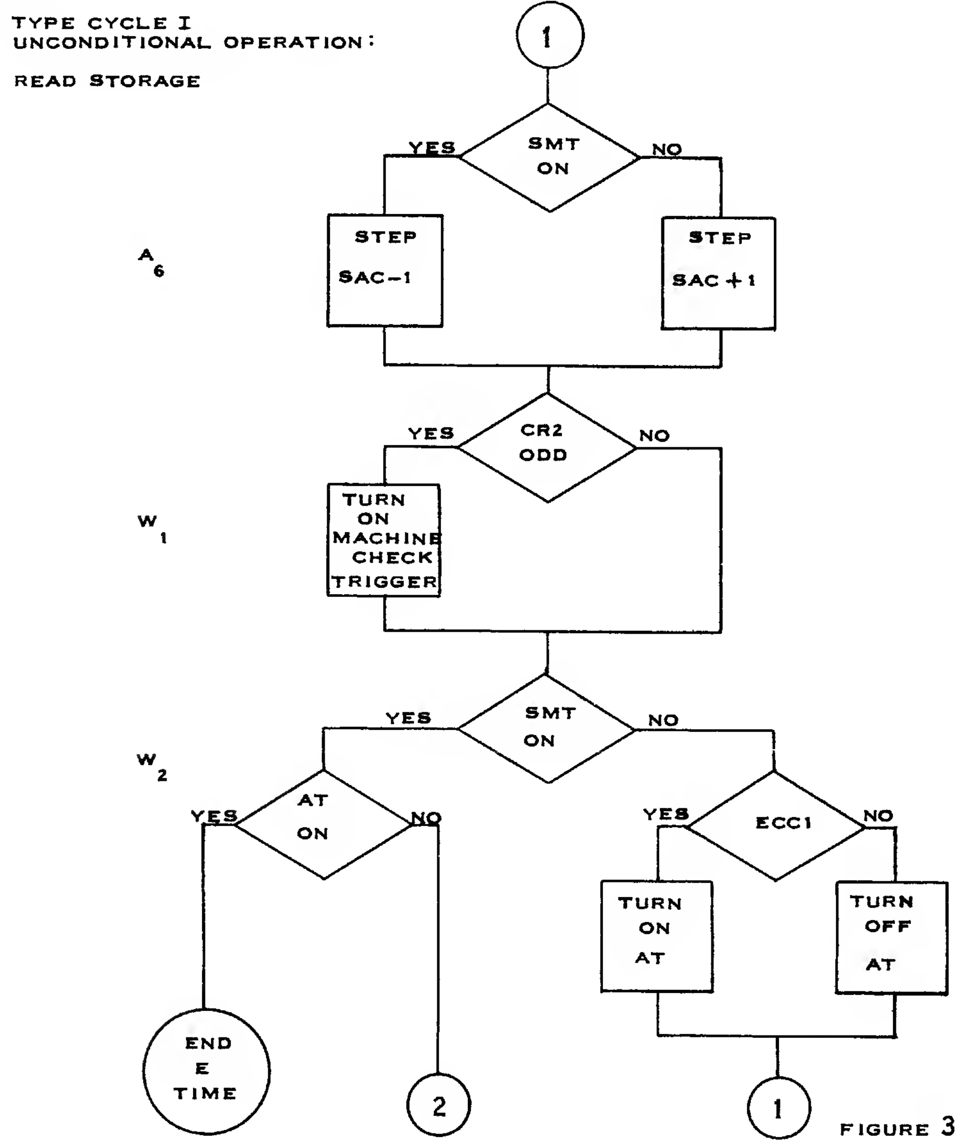


FIGURE 38

NTR

TYPE CYCLE II
UNCONDITIONAL OPERATION:

READ STORAGE

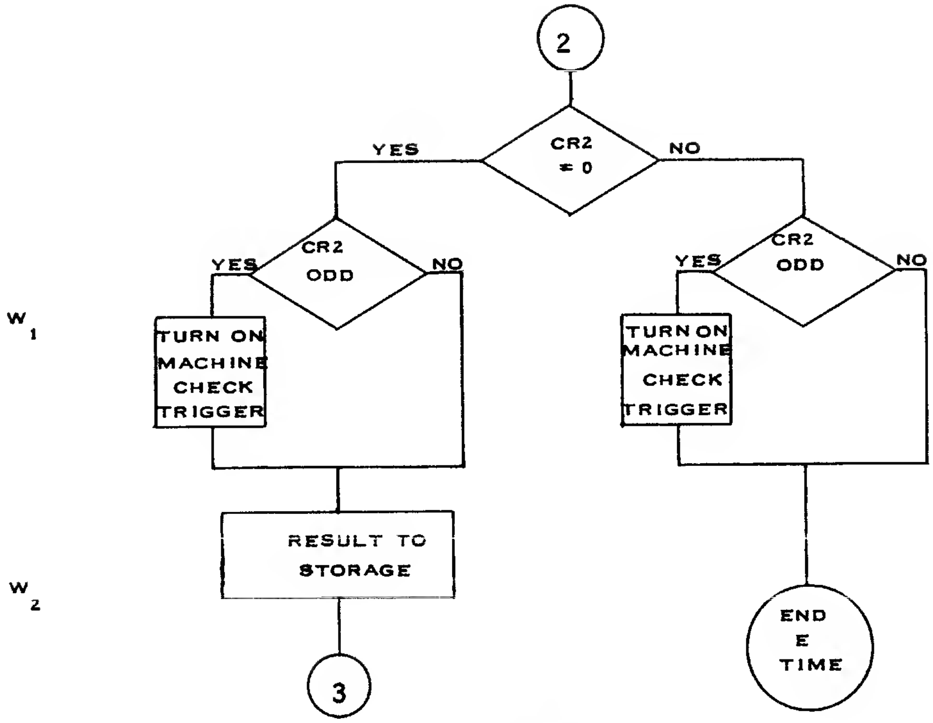


FIGURE 39

NTR

TYPE CYCLE III

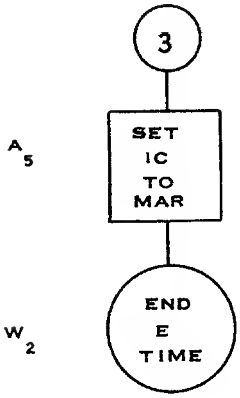


FIGURE 40

SET

During ECC1 of the SET instruction (Figure 41), the DZT is turned on in the same way as in the LOD instruction. In addition, MAC I is stepped minus 1 in view of the fact that the MAC I Equals Zero Trigger is turned on when MAC I actually shows 19999. Therefore, it is necessary to step MAC I down once independently in order to set the proper number of places. This is true in all instructions where MAC I is used as a counter.

After ECC1, a zero is emitted to the Result Register each cycle but is only placed in storage if the SMT is On. Otherwise, the original character is returned to storage by the Storage-In Switch because the Result Register is not sent to storage. A CR2 code check is only made if the SMT is Off. After the SMT is turned On, zeros will be placed in storage to replace the characters previously erased; therefore, no provision is made for checking these characters. If the DZT is still on after MAC I has reached zero, the storage sign is set to plus.

When a 0901 is detected in the SET, SHR, or RND instructions because of a redundancy in the storage unit selected, both the CR1 and CR2 check lights on the console will be turned on. There is no CR1 redundancy, as memory is not read in these instructions.

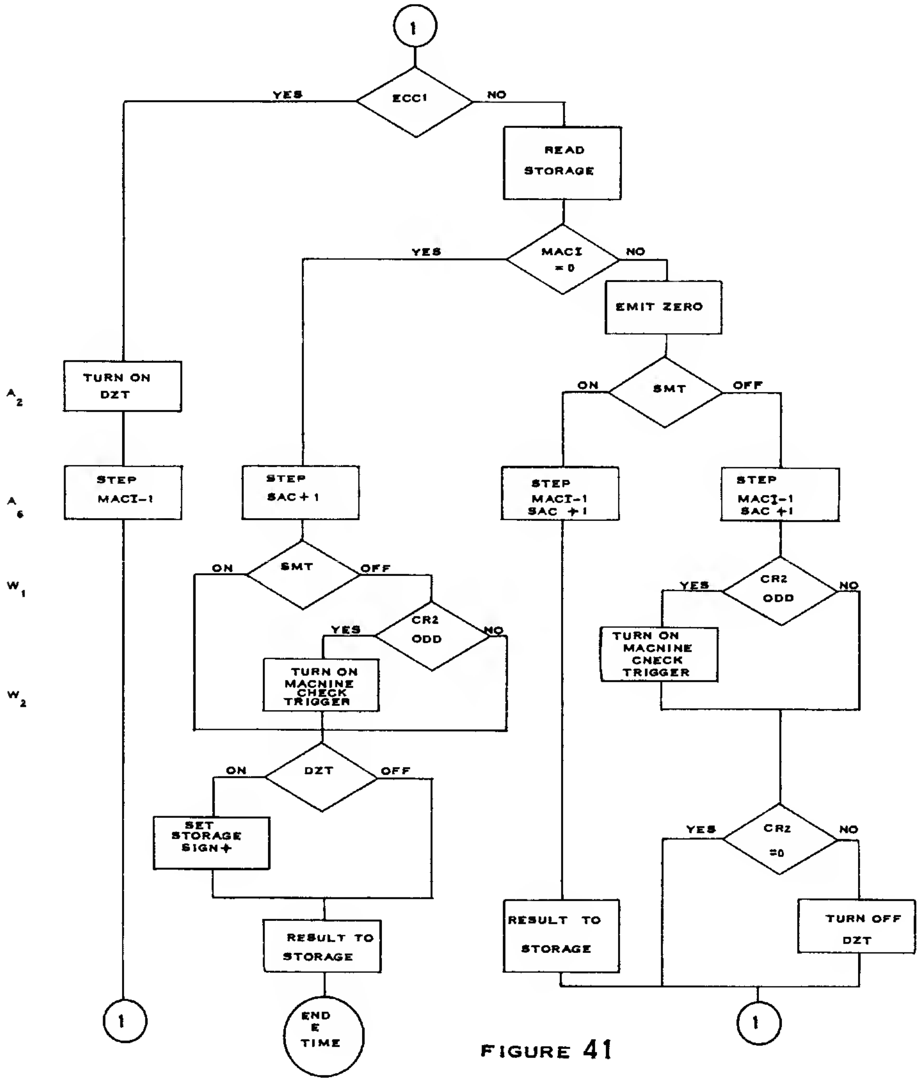


FIGURE 41

LNG

During ECC1 of the LNG instruction (Figure 42), MAC I is stepped minus 1 so that zero can be properly represented by 19999, and SAC is also stepped minus 1. (During I-time, SAC is always set to either SPC or SSR, depending on whether SSR is equal to zero or not). Therefore, during execution of the LNG instruction, SAC and SPC are one position apart. After ECC1 has been completed, MAC I, SAC and SPC are stepped together and the storage position indicated by SAC is read in order to erase the character contained in storage at that time. As long as MAC I is unequal to zero, zeros are emitted into the Result Register and sent to storage. When MAC I equals zero, a zero is not emitted. Accordingly, since the Result Register is still sent to storage, a storage mark is entered to the right of the SPC. A storage mark is entered to the right of the SPC even if the instruction is LNG 0000.

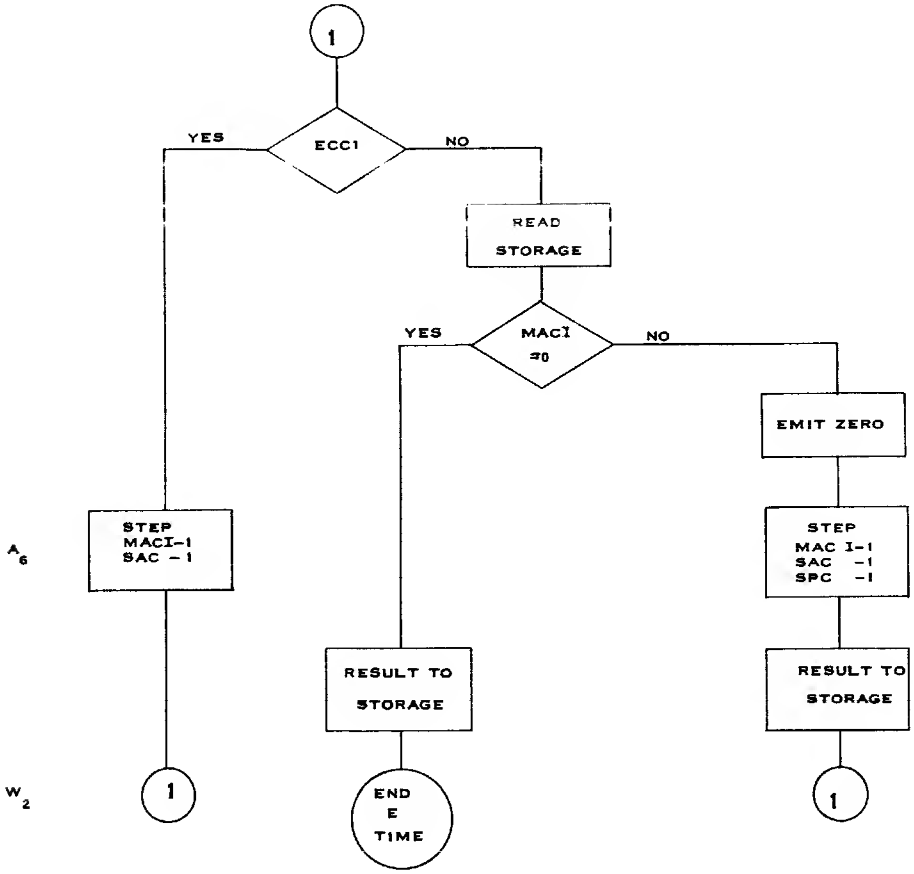


FIGURE 42

SHR/RND

The SHR and RND instructions (Figures 43, 44, 45, and 46) use much of the same circuitry. Consequently, the logic of both instructions is shown together. There are three types of cycles used with these instructions: the RND instruction uses all three of these cycles, whereas the SHR instruction uses only Type Cycles I and III. Type Cycle I has as its function to step down MAC I, SAC, and SPC until MAC I equals zero. Then, the RND instruction enters Type Cycle II, where the half adjustment is effected, provided the instruction was not RND 0000. If the instruction is, however, RND 0000, Type Cycle III is entered directly from Type Cycle I. In Type Cycle III, any part of the field not examined previously is interrogated for a non-zero character, the sensing of which turns off the DZT.

In Type Cycle I, the AT is used to determine whether the instruction was RND 0000. The AT is turned on during ECC1 and turned off in the next cycle if MAC I is not equal to zero. (As mentioned previously, the MAC I Equals Zero Trigger is turned on when MAC I shows 19999). During a RND instruction, the status of the AT determines whether Type Cycle II or III should be entered. SAC must be stepped minus 1 when the MAC I Equals Zero Trigger has been turned on in order to return to the number which is to be increased by five.

During Type Cycle II, the field in storage is half adjusted. The AT is used to indicate the first character cycle of Type Cycle II during which the half adjustment occurs, i.e., 5 is added to the last digit excluded from the field. If a storage mark is sensed before all the carries have been propagated during a rounding operation, the overflow check trigger is turned on. Note, also, that if a RND instruction is given with the accumulator set to zero, the overflow trigger is activated. Type Cycle III is entered when all the decimal carries have been absorbed.

The overflow check trigger will stop the machine at the end of the cycle following that cycle in which it is activated. The overflow check trigger is turned on at W_2 time and the Stop trigger, at W_1 time; consequently, the machine must wait for the following type cycle before the Stop trigger is activated. Because of this cycle delay, all RND instructions will be properly completed even if an overflow occurs.

During Type Cycle III the rest of the field in the accumulator is examined for zeros. The Off condition of AT is used to indicate an overflow in this type cycle. If the AT is Off, storage is read and the Result Register sent to storage, which causes a storage mark to be entered in the proper position. If the AT is On, then the field is examined until a non-zero character is found, at which point the DZT is turned off and E-time is ended. If a non-zero character is not found before the storage mark is reached, the storage sign is set to plus and E-time ended.

SHR/RND

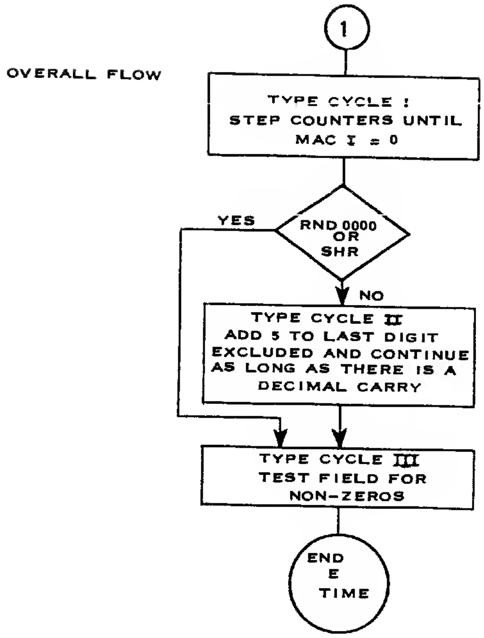


FIGURE 43

SHR/RND

TYPE CYCLE I

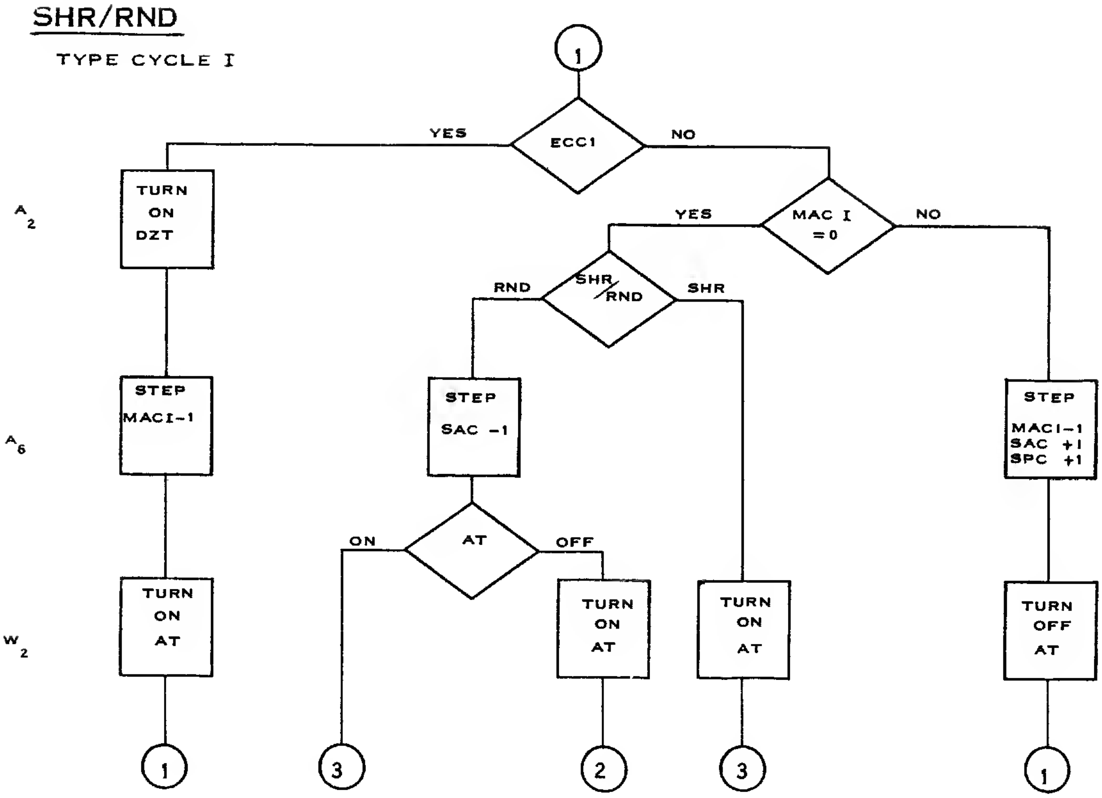


FIGURE 44

SHR/RND

(ACTUALLY ONLY USED BY THE RND INSTRUCTION)

TYPE CYCLE II
UNCONDITIONAL OPERATIONS:

READ STORAGE
RESULT TO STORAGE
CR2 TO TC
C-BIT GENERATOR OUT
DIGIT ADDER OUT

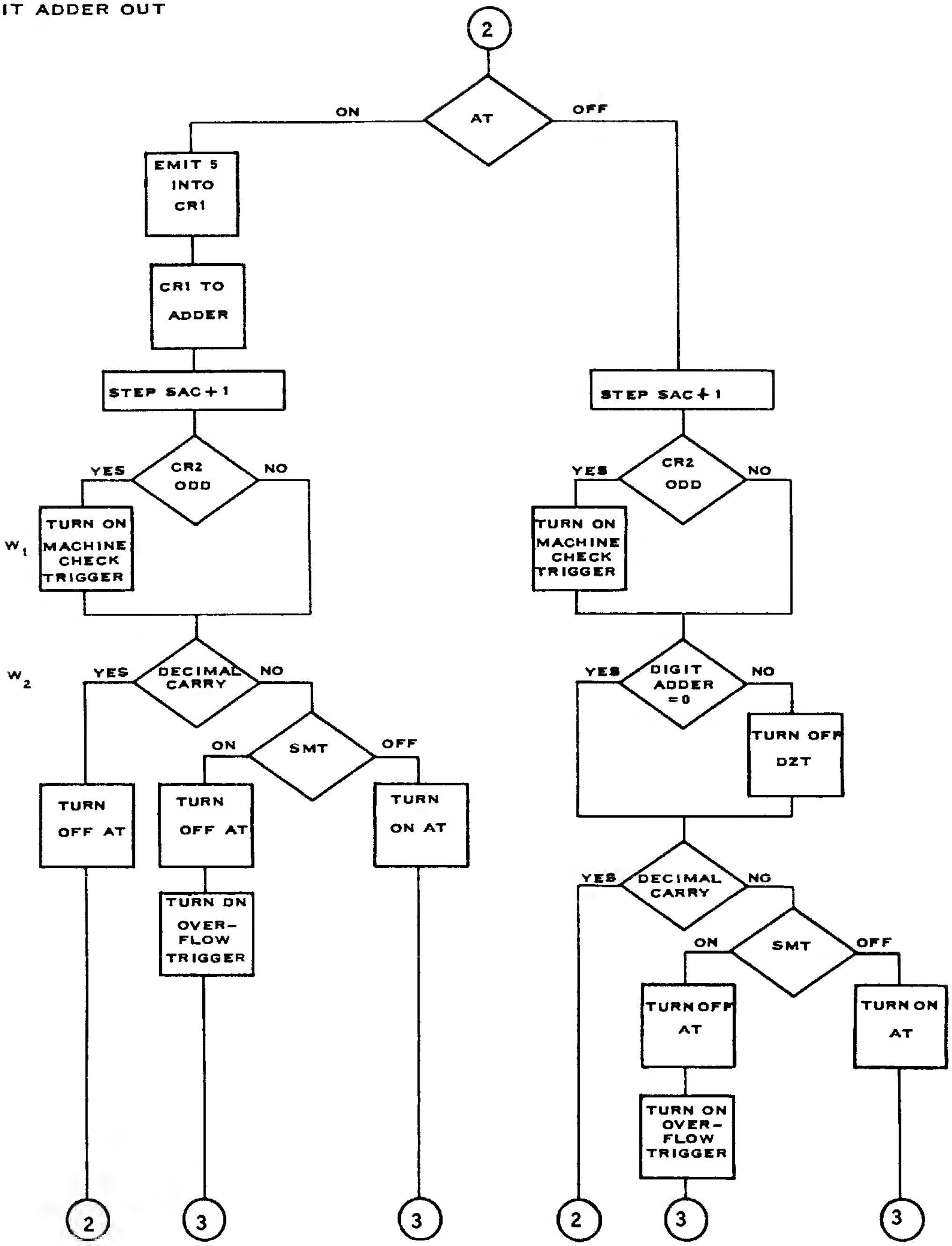


FIGURE 45

SHR/RND

TYPE CYCLE III
UNCONDITIONAL OPERATION:

READ STORAGE

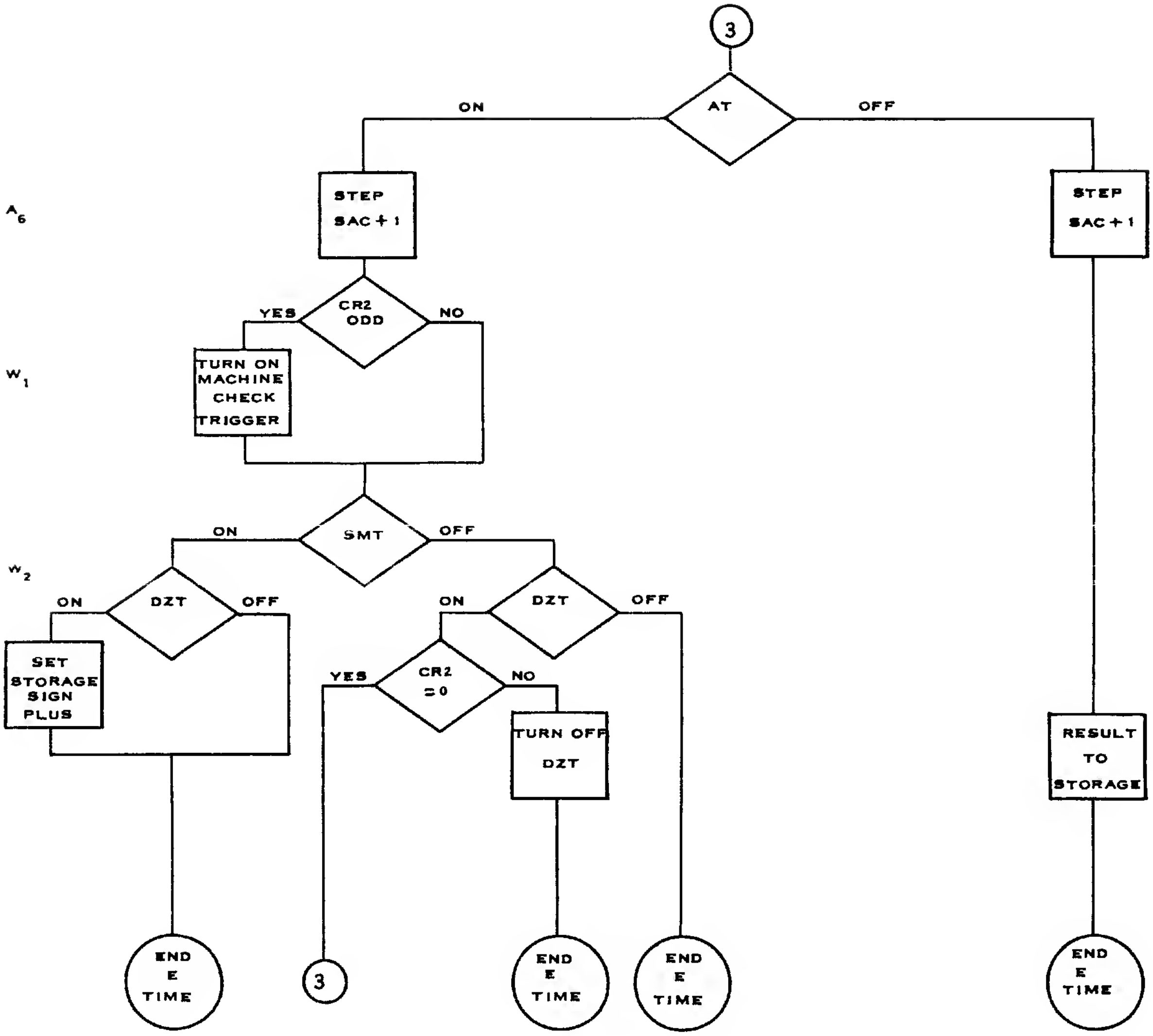


FIGURE 46

ADD/SUB/RAD/RSU

The above four instructions, to all of which Figures 47 through 50 apply, use basically the same circuitry, in much the same way as the SHR and RND instructions explained before.

The function of Type Cycle I is to add the relevant memory and storage fields algebraically until the end of both fields is reached. As usual, during ECC1 the DZT is turned on and is subsequently turned off only if the output of the digit adder is a non-zero character. The Memory Sign Trigger is set to minus if CR1 is minus. This trigger is set to plus every I-time and, unless actually set to minus, always remains at plus. In these instructions, the contents of CR1 or CR2 or both are sent to the digit adder until the memory and storage fields are exhausted. The C-Bit Generator operates in order to provide C-bits, where needed, for the new character created by the addition. During ECC1, the units position of the field in memory addressed by the arithmetic instruction is checked for plus or minus zoning, the lack of which will cause the Sign Check indicator to be turned on.

In the RAD and RSU instructions, the SMT is turned on during ECC1 which means that the contents of CR2 will not reach the digit adder, thereby allowing the characters from memory to pass through the digit adder without the addition of the characters from CR2. Also, if the instruction being operated on is RAD, the storage sign will be set at W_2 time, to the sign indicated by the memory sign trigger. The only difference between the RAD and the RSU instructions is that, at W_2 time, the storage sign becomes the opposite to the one indicated by the memory sign trigger.

Another trigger set during ECC1 is the add/subtract trigger, which does not refer to the ADD or SUB instructions but to the resulting addition or subtraction of absolute values. In the case of an ADD instruction and unlike signs, or a SUB instruction and like signs, this trigger is set to "subtract". The add/subtract trigger is set to "add" every I-time and remains in that status, unless actually set to "subtract". Thus, it parallels closely the mode of operation of the memory sign trigger. When the add/subtract trigger is set to "subtract", it causes the TC to complement the field in storage and also produces a decimal carry-in during ECC1. This procedure will generate the 10's complement of the number rather than the 9's complement, which is the usual output of the TC. (The 9's complement of any number plus one equals the 10's complement of that number.)

When the first non-numerical character is read from memory, a trigger called Memory End Trigger (MET) is turned on. This check is not made during ECC1, which prevents the signed digit in the units position of the field from turning on the MET. After ECC1, Type Cycle I continues until both the SMT and MET are turned on. At this point, one of the following four situations will arise:

1 No decimal carry with the add/subtract trigger set to "add"

Two pertinent examples are shown below. Note that if the DZT is On, the storage sign would be set to plus before E-time is ended.

Memory	b8$\overset{+}{1}$	b02$\overset{+}{4}$
Storage	+ @11	+ @38
	+ @92	+@062

2 A decimal carry with the add/subtract trigger set to "subtract"

In this case, illustrated by the first example below, the presence of the decimal carry indicates that the memory field is larger than the storage field; therefore, the sign of the storage field must be reversed. In this situation, the decimal carry is lost since the contents of the digit adder are not routed to the Result Register. However, the Result Register is still sent to storage, causing a storage mark to be created. Unlike the RAD and RSU instructions, the sign of storage is not disturbed in the ADD and SUB instructions until the end of both fields is reached. If the DZT is still On, the storage sign is set to plus regardless of its previous setting, as illustrated by the second example.

Storage	-@81	-@75
Memory	b8$\overset{-}{2}$	b7$\overset{-}{5}$
Storage Complemented	-@19	-@25
	-@01 Carry	-@00 Carry
Final Result	-@01	+@00

3 A decimal carry with the add/subtract trigger set to "add"

This is the overflow condition, an example of which is shown on the following page. (When the add/subtract trigger is set to "subtract", overflow is impossible, for the absolute difference between two numbers cannot be greater than the larger of the two numbers.) When an overflow occurs, Type Cycle II is entered, which, as its sole function, causes a storage mark to be placed to the left of the new field.

Memory		$b8\overset{+}{1}$
Storage	+	@20
	+	@101

4 No decimal carry with the add/subtract trigger set to "subtract"

Since this condition arises only when the memory field is smaller than the storage field, the sign of storage is never disturbed. The two examples shown below illustrate that the complement of the correct answer is developed during Type Cycle I. In this case, Type Cycle II is entered, where the storage field is recomplemented, to produce the correct answer. The AT is used in Type Cycle II to produce a decimal carry-in during the first cycle to generate the 10's complement.

Storage	+@58	-@33
Memory	$b2\overset{-}{1}$	$b3\overset{+}{2}$
Storage Complemented	+@42	-@67
End of Type Cycle I	+@63	-@99
Recomplemented	+@37	-@01

ADD/SUB/RAD/RSU

OVERALL FLOW

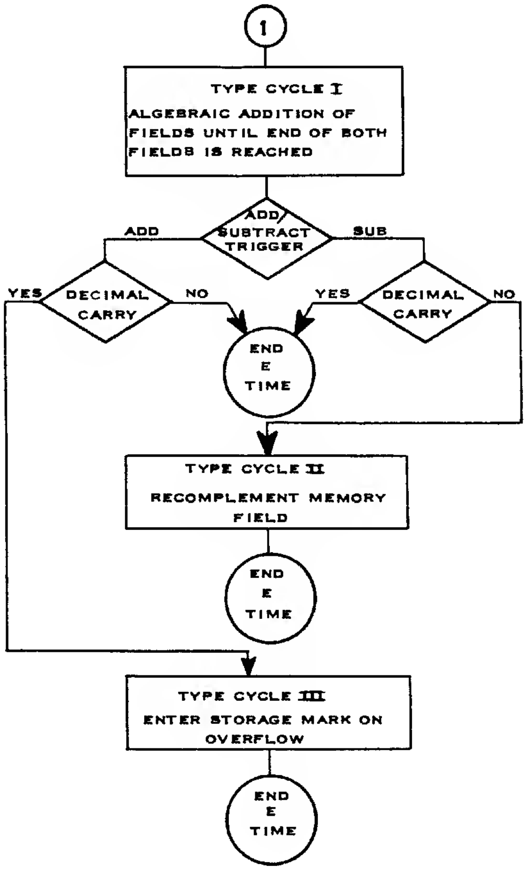


FIGURE 47

ADD/SUB/RAD/RSU

UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
C-BIT GENERATOR OUT
RESULT TO STORAGE

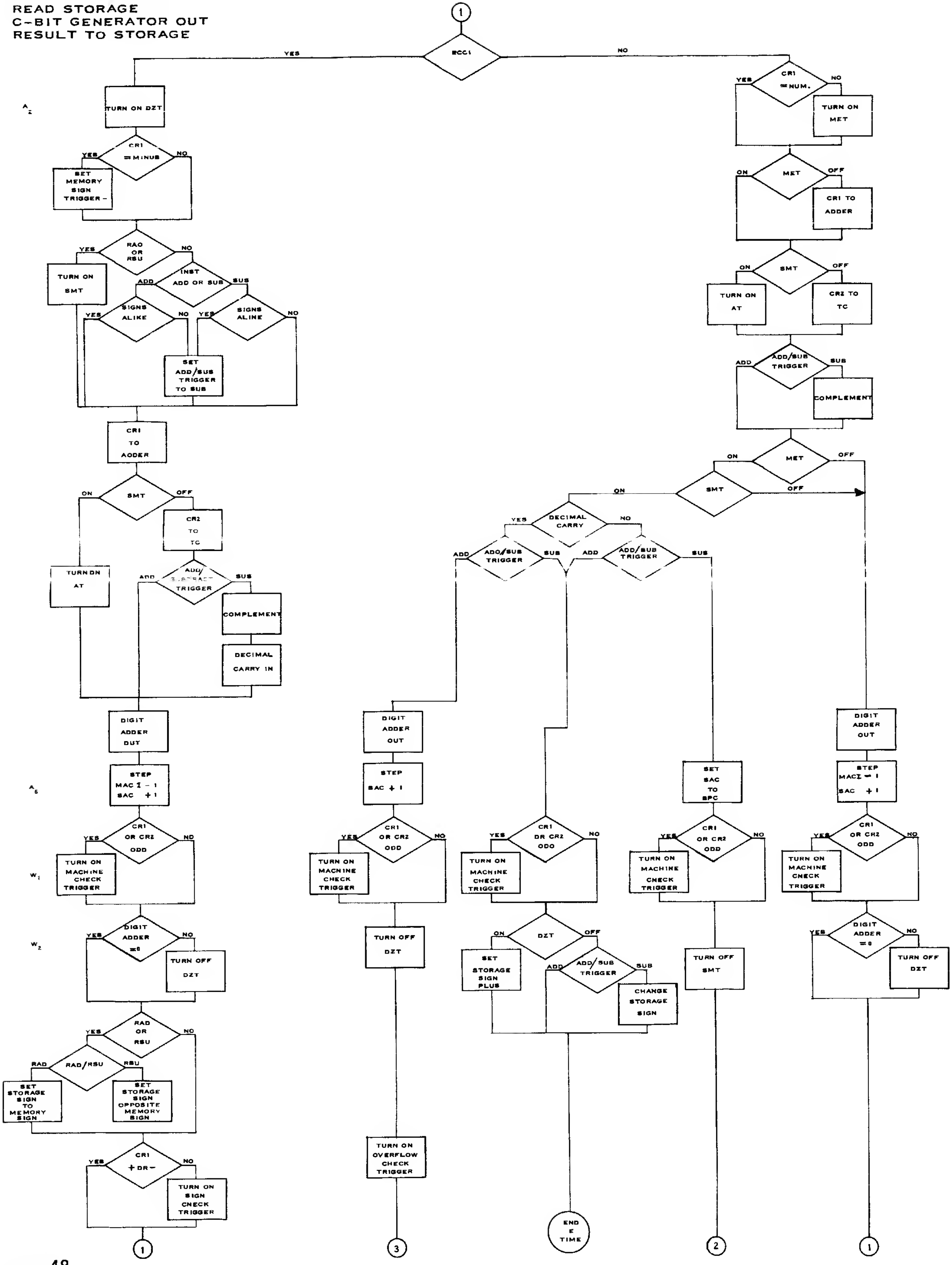


FIGURE 48

ADD/SUB/RAD/RSU

TYPE CYCLE II
UNCONDITIONAL OPERATIONS:

READ STORAGE
CR2 TO TC
COMPLEMENT
C-BIT GENERATOR OUT
RESULT TO STORAGE

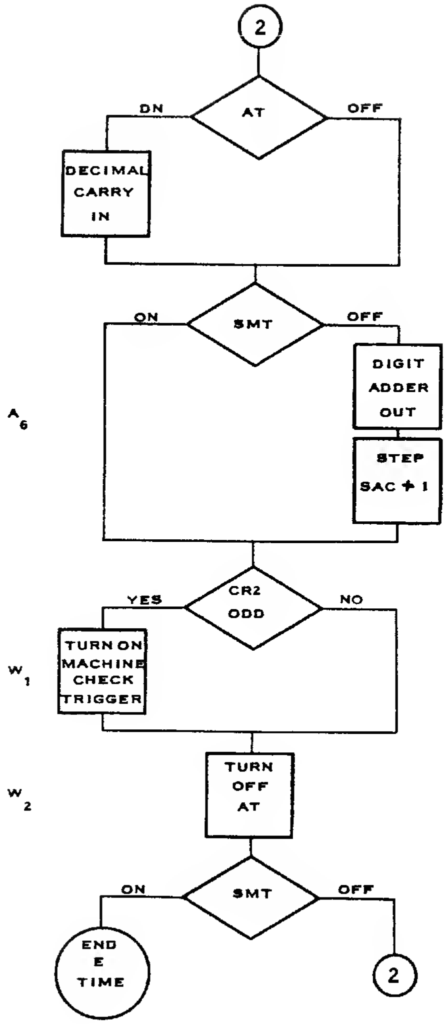


FIGURE 49

ADD/SUB/RAD/RSU

TYPE CYCLE III
UNCONDITIONAL OPERATIONS:

READ STORAGE
RESULT REGISTER TO STORAGE

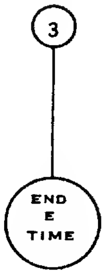


FIGURE 50

CMP

In the CMP instruction (Figure 51) generally, the comparison of each pair of characters is achieved by subtraction to determine which is the larger or whether they are equal.

Both memory and storage are read and sent to the adder. CR2 goes through the TC which, in this case, is signaled to complement. Bringing up the complement line also causes the true complementer of the zone adder to complement so that the zones of the characters can be properly subtracted.

Since the characters are subtracted in individual pairs rather than as two fields in algebraic subtraction, the decimal carry-in is provided in each cycle in order to provide the proper 10's complement for each character from storage without regard to previous carries or non-carries. In addition, a zone carry-in is provided in each character cycle, as the true complementer of the zone adder operates in a manner similar to the true complementer of the digit adder.

When the contents of CR1 are greater than 10, the 8 bit does not enter the adder, which is the case when special characters, such as a dollar sign ($), are compared. For characters from storage, the TC is so designed that if the numerical portion of a character larger than 10 is complemented, complementation is accomplished without reference to the 8 bit. The complement of the numerical portion (12) of an asterisk (*), for example, is 5, which is the 9's complement of 4, or 12 minus 8 (as the 8 bit is not complemented).

At the beginning of ECC1, the equal trigger is turned on and the high trigger is turned off. As long as SMT is Off, both MAC I and SAC are stepped.

A series of tests are then made to determine if either CR1 or CR2 contains a blank or a special character. (A special character, for the purpose of the CMP instruction, is defined as any character in the collating sequence up to the commercial "at", @, other than a blank.) If an inequality is found, the equal trigger is turned off and will not be restored until ECC1 of the next CMP instruction. If, then, storage is low, the high trigger is turned off; if, however, storage is high, this trigger is turned on. The high trigger may be turned on and off several times in a CMP instruction because of the manner in which the 705 compares.

If CR1 and CR2, both do, or do not,contain special characters, the output of the zone adder and digit adder must be examined to determine if the characters are equal or not. If the output of both adders is zero, then no triggers are distributed and the 705 proceeds to compare the next two characters. If the output of either the zone adder or digit adder is unequal to zero, then an inequality has been found and the equal trigger is turned off.

A field is compared, character by character, from right to left. For example, if "802" is the field in memory and "652" the field in storage, the high trigger is turned on when the tens position is compared but is subsequently turned off in the third ECC, when the hundreds position is compared. Whenever a storage mark is read, E-time is ended without disturbing either the high or equal trigger.

In the CMP instruction, the zone adder takes precedence over the digit adder; if characters have different zones, the numerical portion has no bearing on whether storage is higher or lower than memory. Zones are collated in the reverse order of digits: a 00 zone is highest in the collating order of zones and an 11 zone, the lowest; whereas a 9 is the highest and 0 the lowest in the collating sequence of digits. Special provision is made to collate a hyphen and ampersand higher than any other special character with the same zoning.

The comparison of two unequal zones is illustrated below. Note that the 4's binary complement is used in the case of zones. Actually, this is very similar to the TC of the digit adder in that the 3's complement is always produced and, then, a one is added to this complement by means of a zone carry. Since zones are compared in the opposite order from digits, a non-zero condition in the zone adder with a carry has the opposite effect on the high trigger from a non-zero condition in the digit adder with a carry.

Memory Zone	10	01
Storage Zone	01	11
Storage Zone Complement	11	01
Total	C01	10

Zone Adder ≠ 0	Zone Adder ≠ 0
Carry	No Carry
Turn on high trigger	Turn off high trigger

CMP

UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CR1 TO ADDER
CR2 TO TC
COMPLEMENT
DECIMAL CARRY IN
ZONE CARRY IN

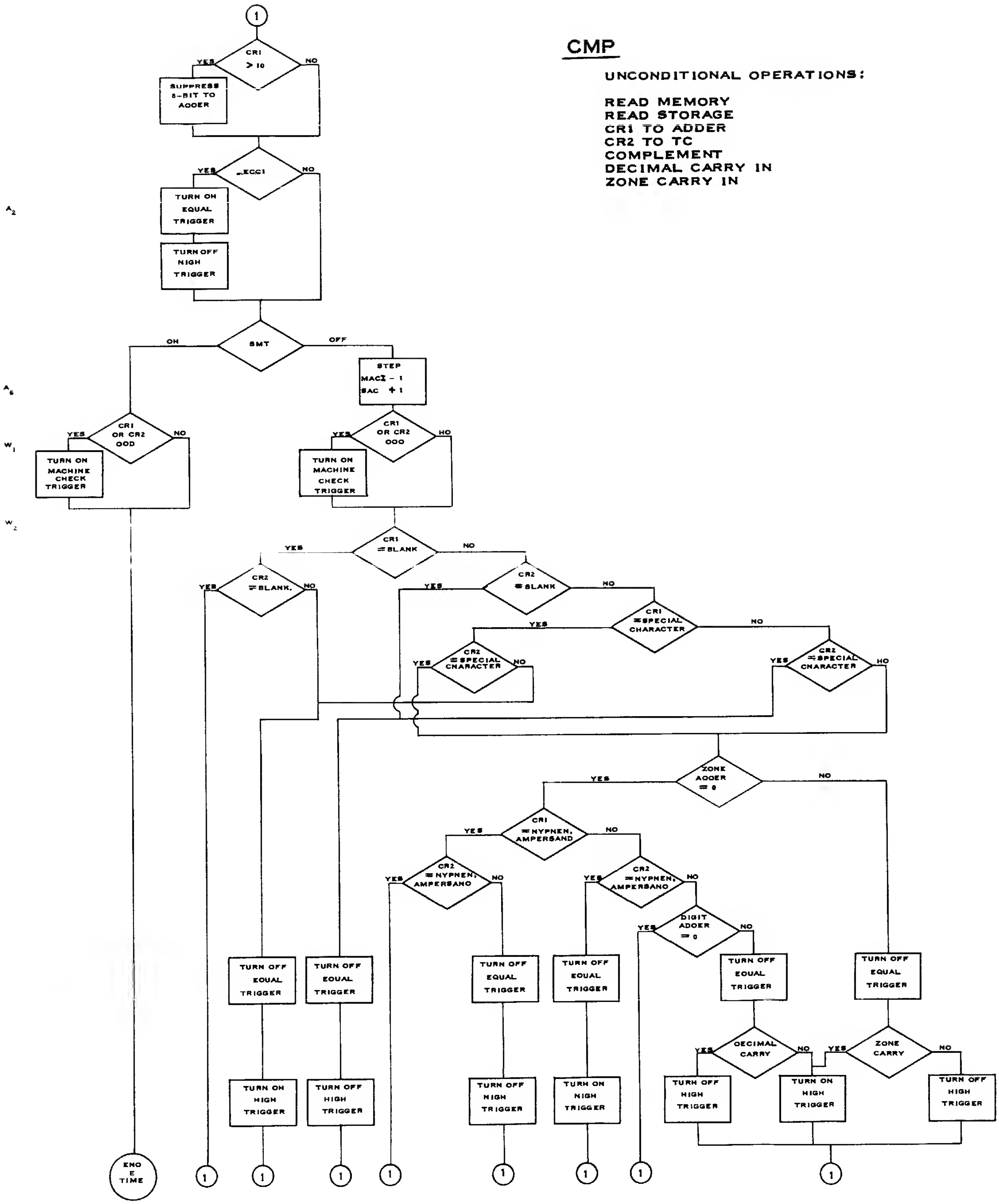


FIGURE 51

SPR

The SPR instruction (Figures 52, 53, and 54) consists of two type cycles. In Type Cycle I the characters are placed in memory from storage and in Type Cycle II any zeros or commas just stored in memory are erased from left to right until a decimal or non-zero character is found, at which point E-time is ended.

During ECC1 of Type Cycle I, a blank or a hyphen is emitted into the units position of the field addressed depending on the setting of the storage sign, plus or minus, respectively. The character emitter emits only a B-bit for a hyphen; therefore, it is also necessary for the C-Bit Generator to supply a C-bit. MAC I stepped down and a redundancy check is made on the character replaced in memory. During the other cycles of Type Cycle I, storage is read, the contents of storage are passed through the zone adder and digit adder, and sent to the Result Register. Unlike the ST instruction, redundancies cannot be created by the SPR instruction because the zone adder is always sent to the Result Register. However, since the digit adder is also routed out every time, the bit structure of a blank, hyphen or ampersand may be altered to that of a record mark, minus zero or plus zero because the digit adder produces a zero (8 and a 2 bit) if it receives no input.

Whenever CR1 contains a period or comma, that character is skipped and the character from storage is stored in the next position in memory. When a storage mark is read, MAC I is stepped back one position so that it will indicate the proper location to examine the memory field, from left to right, when Type Cycle II is entered.

In Type Cycle II the memory field is tested, from left to right, for zeros or commas. If either of these characters is present, the Result Register, into which a blank has been emitted, is sent to memory and the next character of the memory field is examined. Whenever a character other than a zero or comma is sensed, contents of the Result Register are not moved to memory; therefore, the character read from memory will be returned, and E-time is ended.

OVERALL FLOW

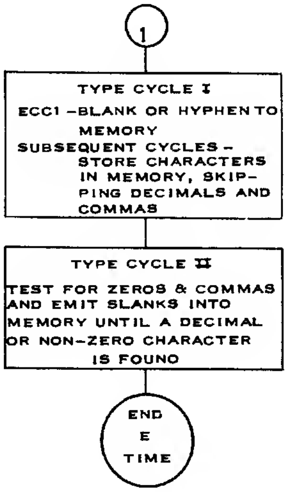

1
TYPE CYCLE I
ECC1 -BLANK OR HYPHEN TO MEMORY
SUBSEQUENT CYCLES - STORE CHARACTERS IN MEMORY, SKIPPING DECIMALS AND COMMAS
TYPE CYCLE II
TEST FOR ZEROS & COMMAS AND EMIT SLANKS INTO MEMORY UNTIL A DECIMAL OR NON-ZERO CHARACTER IS FOUNO
END E TIME

FIGURE 52

SPR

TYPE CYCLE I

READ MEMORY

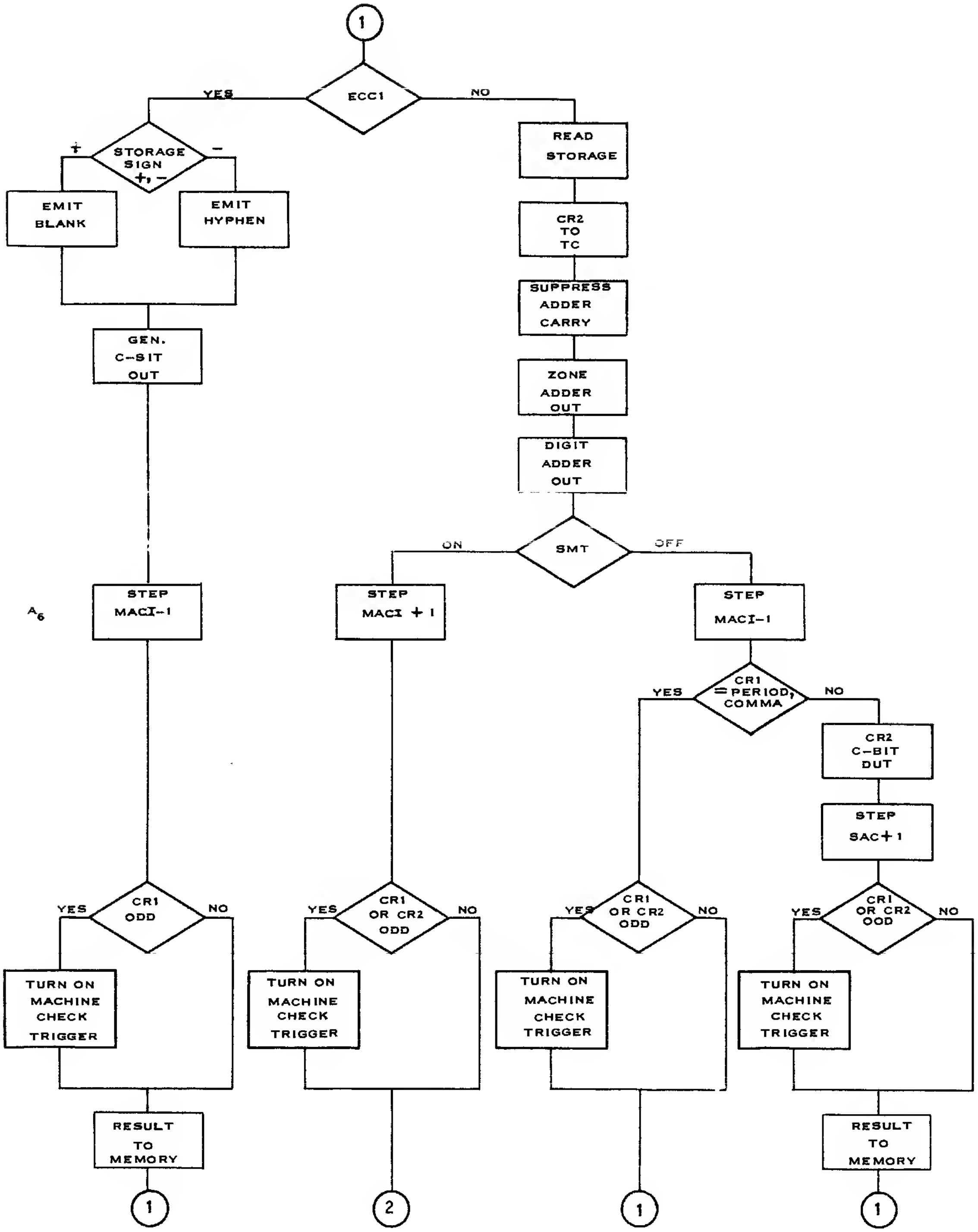


FIGURE 53

SPR

TYPE CYCLE II
UNCONDITIONAL OPERATIONS:

READ MEMORY
EMIT BLANK

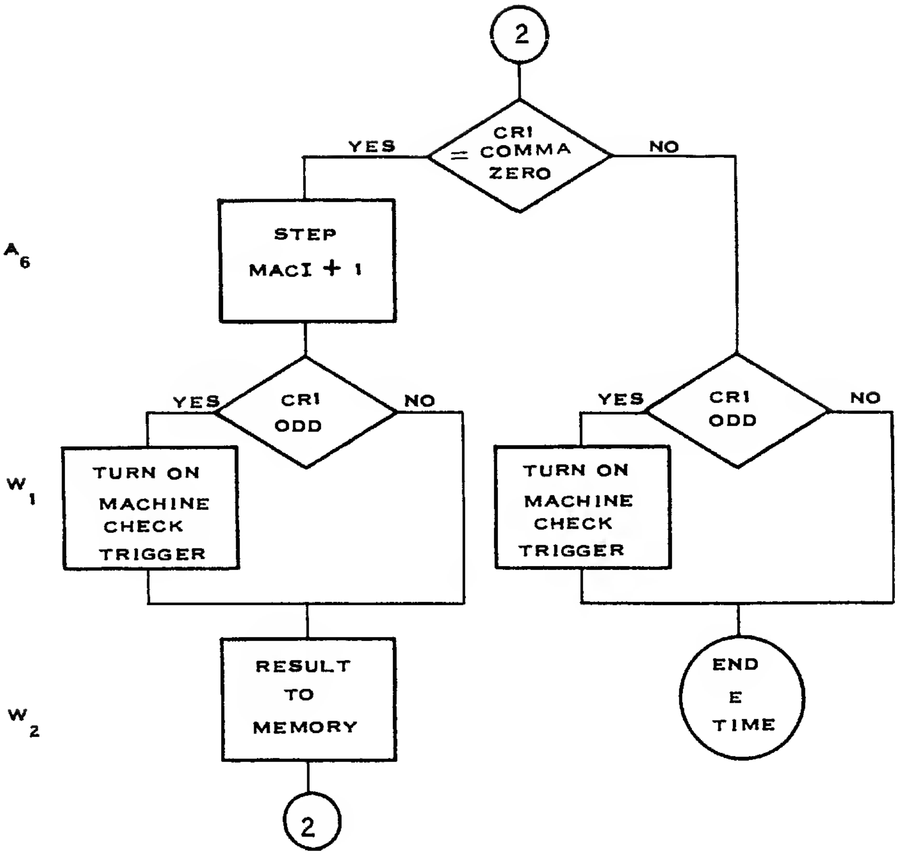


FIGURE 54

TMT

The TMT instruction (Figures 55, 56, and 57) is unique in several respects. In order to move a single character or a block of five characters in a TMT operation, two character cycles are required. However, since no arithmetic operations are performed, the arithmetic portion of the character cycle is eliminated, and each cycle, therefore, becomes 9 microseconds in length. The TMT operation alternates between Type Cycle I and II until the limiting factor of a storage mark or record mark, depending on whether it is serial or high-speed transmission, is reached.

In Type Cycle I, the address governing the reading of memory is the address of the TMT instruction as modified by the stepping of MAC I in Type Cycle II. In any instruction, at the beginning of each execution character cycle, MASR is set to MAC I unless it is specifically set to MAC II. This also holds true in the case of the TMT instruction and causes MASR to be set to MAC I every time Type Cycle I is entered. The character or characters to be transmitted are read from memory and then placed in the MBR and CR1. In a high-speed TMT instruction, the character in the 4 or 9 position* goes to CR1; characters are normally moved into CR1 or CR2 at A_0 time, but as there is no A_0 time in a R-W cycle, this is accomplished here in W_0 time. If SSR is unequal to zero, i.e., if serial transmission is indicated, the ASU designated is read. In high-speed transmission, however, storage is not referred to in any way.

Memory positions operated on in Type Cycle II are indicated by MAC II prior to its being stepped later in the cycle. The functions of this type cycle vary depending on whether SSR is, or is not, equal to zero. If SSR equals zero, memory is read during the R portion of the cycle, but is not placed in MBR, thereby erasing the five characters at the RCV address, and subsequent higher memory addresses, while the five characters from the TMT address are left in the MBR. On the W portion of Type Cycle II, the characters in the MBR are written into the erased area in memory. If any of these five characters has an odd number of bits, the MBR redundancy check will cause the machine check trigger to be turned on. CR1 has not been reset so that a record mark in a 4 or 9 position can be sensed in Type Cycle II in order to terminate E-time.

If, in Type Cycle II, SSR is unequal to zero, then serial transmission is effected. Normally both CR1 and the MBR are reset every character cycle; the TMT instruction is the only exception: in Type Cycle II CR1 reset is suppressed so that the character to be transmitted will not be erased and no new character is moved into CR1. In serial transmission, only one character at a time is erased from memory rather than five as in high-speed transmission.

*A "4 or 9 position" is a location in memory, the address of which has a 4 or 9 as the units digit.

The character from CR1 is assembled in the Result Register and from there sent to memory, since it is impossible to send a character from CR1 to memory directly. If the character in CR1 is a blank, hyphen or ampersand, the digit adder is not routed out in order to conserve the no-bit structure in the numerical portion of these characters. A redundancy check is made on the characters in both CR1 and CR2. The redundancy check on the character in CR2 may not appear to serve a useful purpose, but it should be noted that, if the storage mark were to pick up an extra bit, transmission may continue indefinitely in the absence of such a redundancy check. Transmission of characters is continued until the storage mark is sensed, then E-time is ended.

TMT

OVERALL FLOW

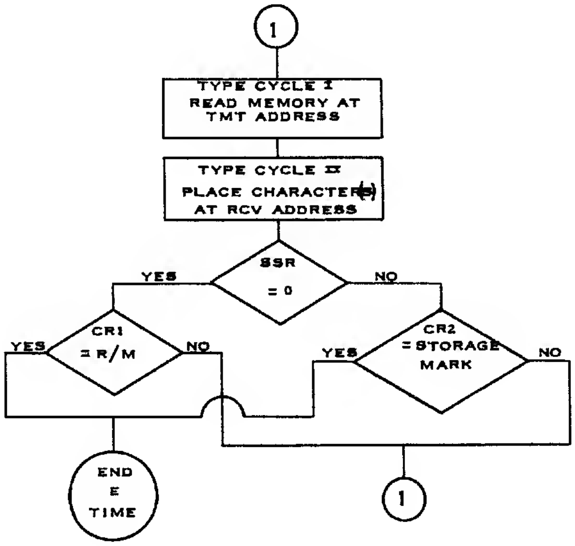


FIGURE 55

TMT

TYPE CYCLE I (R AND W TIME ONLY)
UNCONDITIONAL OPERATION:

READ MEMORY

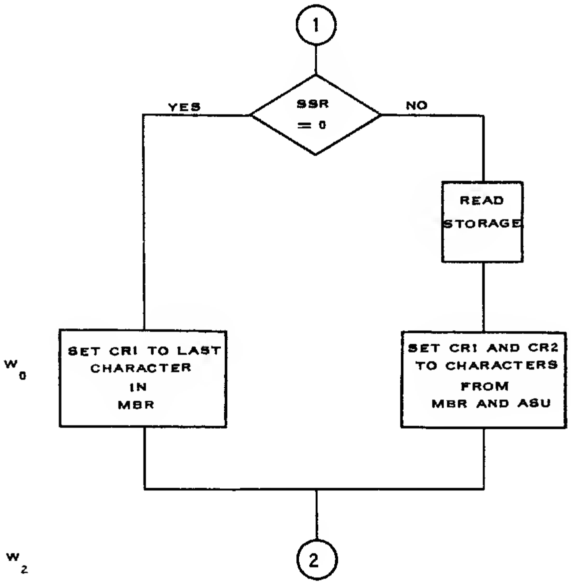


FIGURE 56

TMT

TYPE CYCLE II (R AND W TIME ONLY)
UNCONDITIONAL OPERATIONS:

SET MASR TO MAC II
READ MEMORY
SUPPRESS CR1 RESET

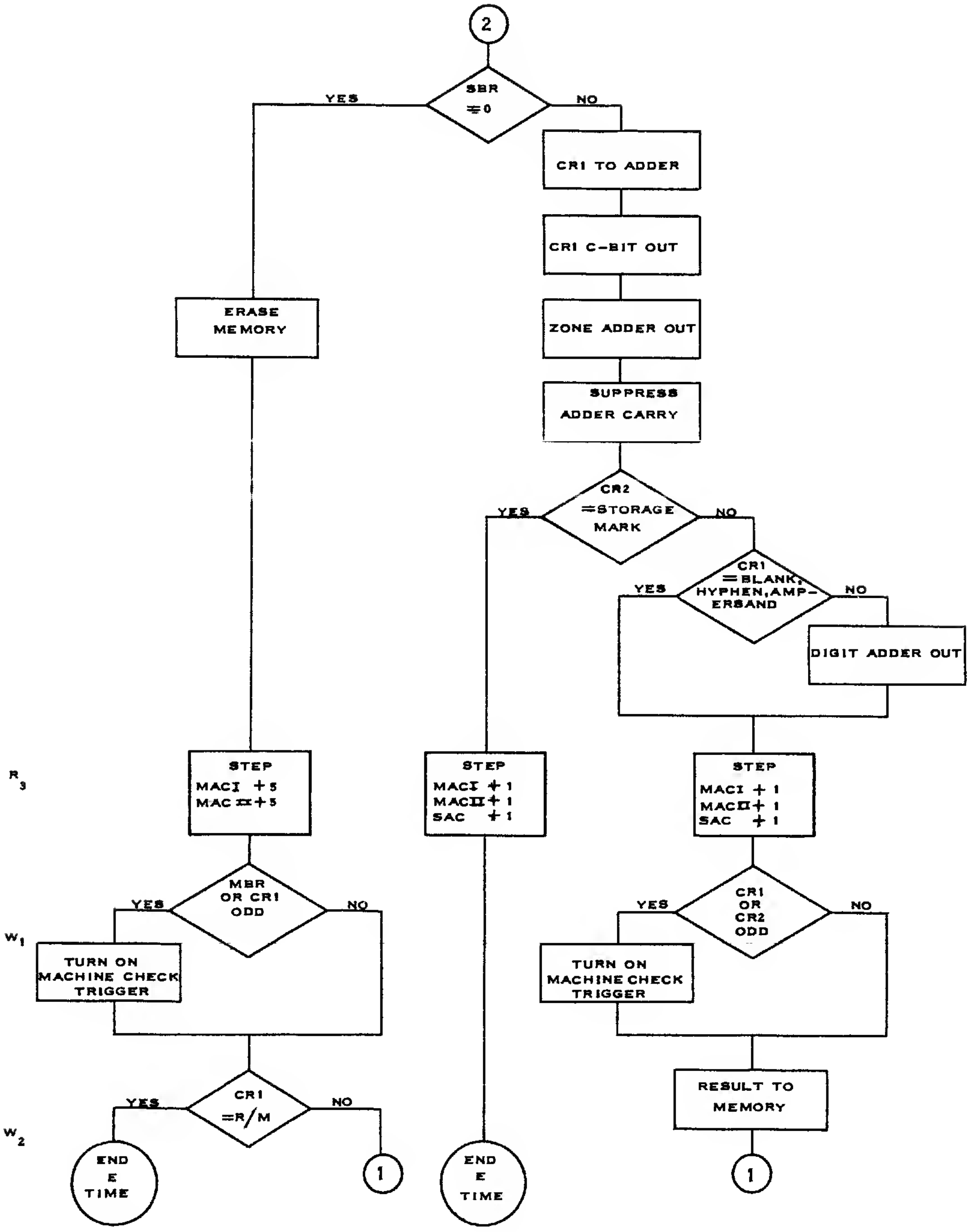


FIGURE 57

ADM

The ADM instruction (Figures 58 through 63) consists of five types of cycles. In Type Cycle I the first characters from both memory and storage are added. If the memory field is unsigned, then the zone adder is sent to the Result Register. If the field is signed, the storage and memory signs are examined to determine if addition or subtraction is to be performed, which are effected in the same way as in the ADD and SUB instructions. In any case, the C-Bit Generator is signaled to operate in order to produce the necessary C-bits for the new characters generated. Type Cycle I lasts only one cycle, after which either Type Cycle II or III is entered, depending on whether the memory field was signed or unsigned.

If the addressed field is signed, Type Cycle II continues the algebraic addition of the fields until the end of the memory field is reached. At this point E-time is normally ended. However, it is sometimes necessary to recomplement the memory field, which is the purpose of Type Cycle IV. This situation occurs when the following two conditions both apply: the storage field is numerically larger than the memory field and their signs are opposite. Type Cycle IV of the ADM instruction is the only case of 705 operation in which the memory field is ever complemented. Here, also, the sign of the memory field is reversed. In Type Cycle IV, the AT is used to indicate the first execution cycle of this type cycle, where a decimal carry-in is provided to create the 10's complement of the relevant memory field.

The following example illustrates the purpose of Type Cycle IV:

Memory $b9\overset{-}{3}$

Storage +@98

Storage Complemented +@02

Result:

End of Type Cycle II $b9\overset{-}{5}$

After recomplementation

End of Type Cycle IV $b0\overset{+}{5}$

When an unsigned field is addressed by an ADM instruction, Type Cycle I is followed by Type Cycle III, where the addition of the fields is continued until the storage mark is reached. If no decimal carry has resulted from the addition of the last two characters, E-time is ended. If there is a decimal carry, however, Type Cycle V, which lasts only one execution cycle, is entered. In Type Cycle V, a zone carry-in is produced which has the effect of adding one to the zone of the last character added. This type cycle thereby accomplishes the zone change associated with an unsigned ADM instruction, having a carry in the memory location corresponding to the high order position of the storage field.

ADM

OVERALL FLOW

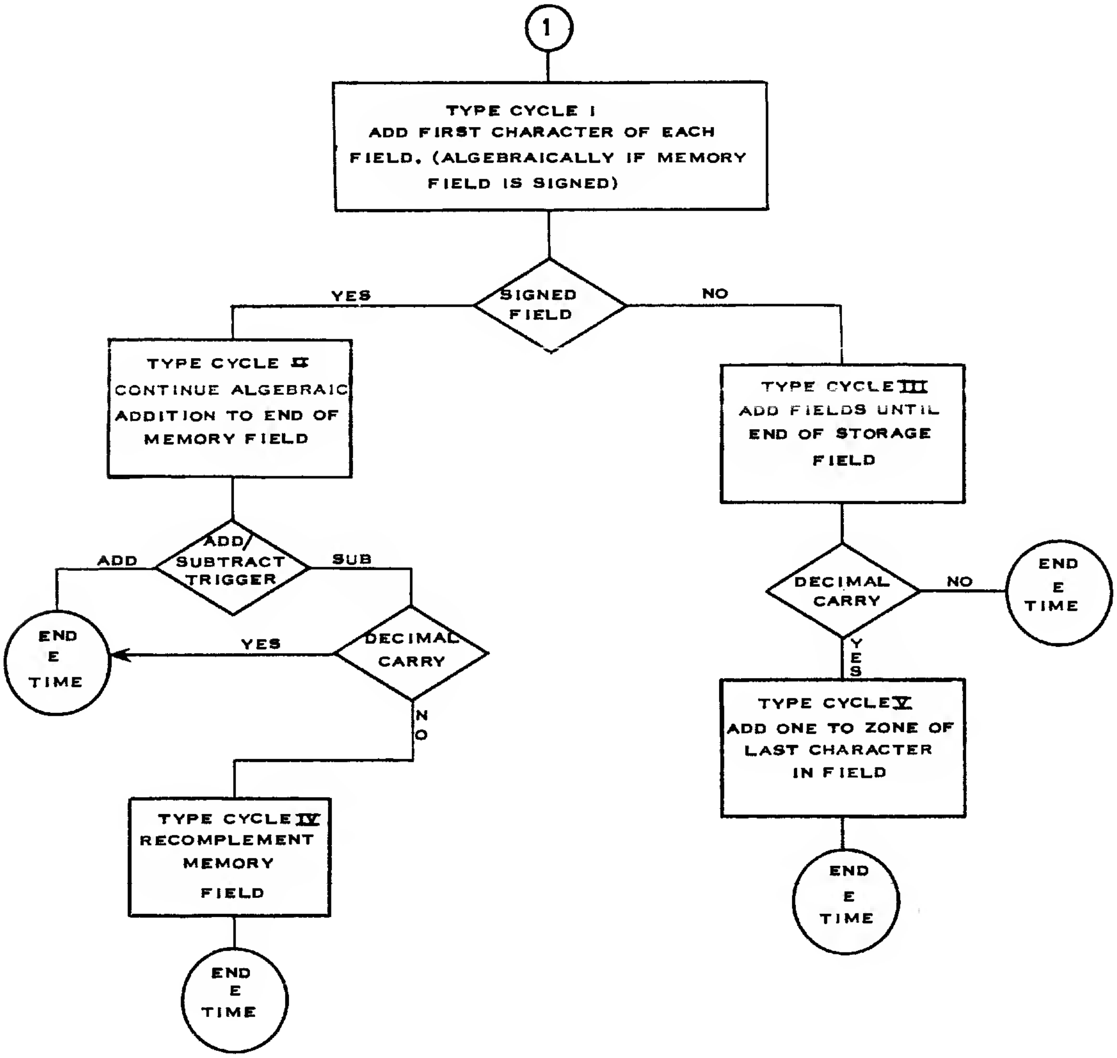

1
TYPE CYCLE I
ADD FIRST CHARACTER OF EACH FIELD. (ALGEBRAICALLY IF MEMORY FIELD IS SIGNED)
SIGNED FIELD
YES
NO
TYPE CYCLE II
CONTINUE ALGEBRAIC ADDITION TO END OF MEMORY FIELD
ADD/ SUBTRACT TRIGGER
ADD
SUB
END E TIME
YES
DECIMAL CARRY
NO
TYPE CYCLE IV
RECOMPLEMENT MEMORY FIELD
END E TIME
TYPE CYCLE III
ADD FIELDS UNTIL END OF STORAGE FIELD
DECIMAL CARRY
NO
END E TIME
YES
TYPE CYCLE V
ADD ONE TO ZONE OF LAST CHARACTER IN FIELD
END E TIME

FIGURE 58

ADM

TYPE CYCLE I
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CR1 TO ADDER
CR2 TO TC
DIGIT ADDER OUT
C-BIT GENERATOR OUT
RESULT TO MEMORY

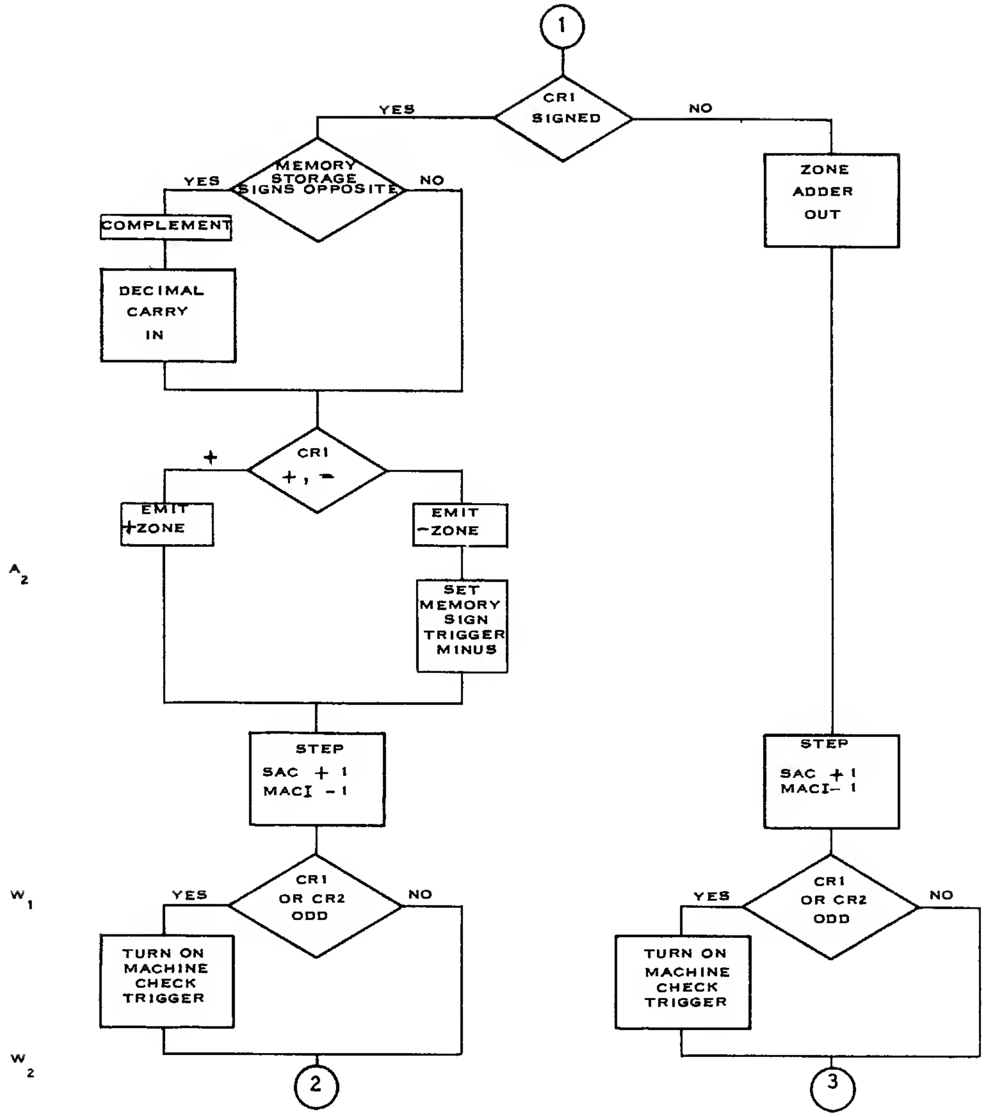


FIGURE 59

ADM

TYPE CYCLE II
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CRI TO ADDER
DIGIT ADDER OUT
C-BIT GENERATOR OUT

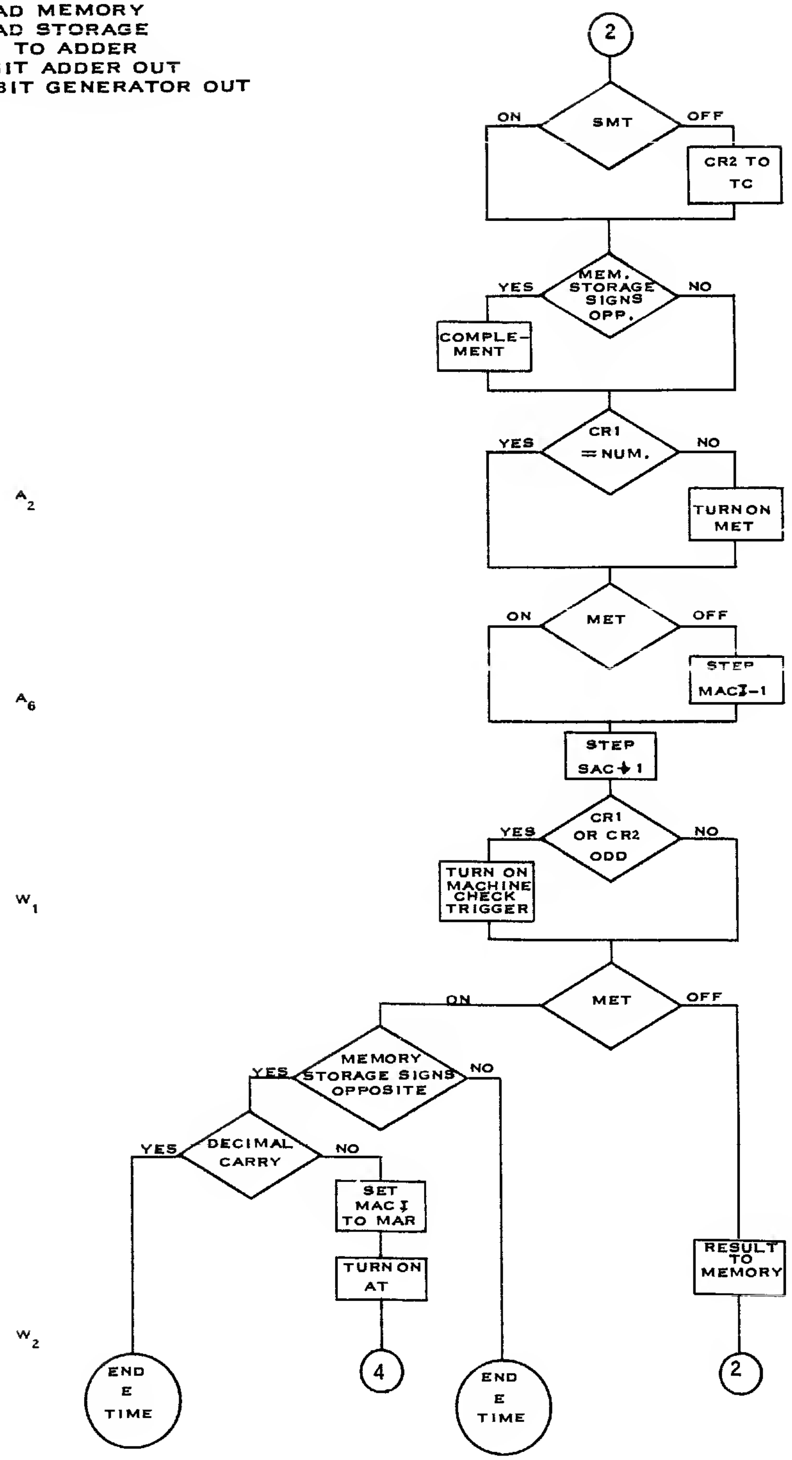


FIGURE 60

ADM

TYPE CYCLE III
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CR1 TO ADDER
CR2 TO TC
DIGIT ADDER OUT
ZONE ADDER OUT
C-BIT GENERATOR OUT

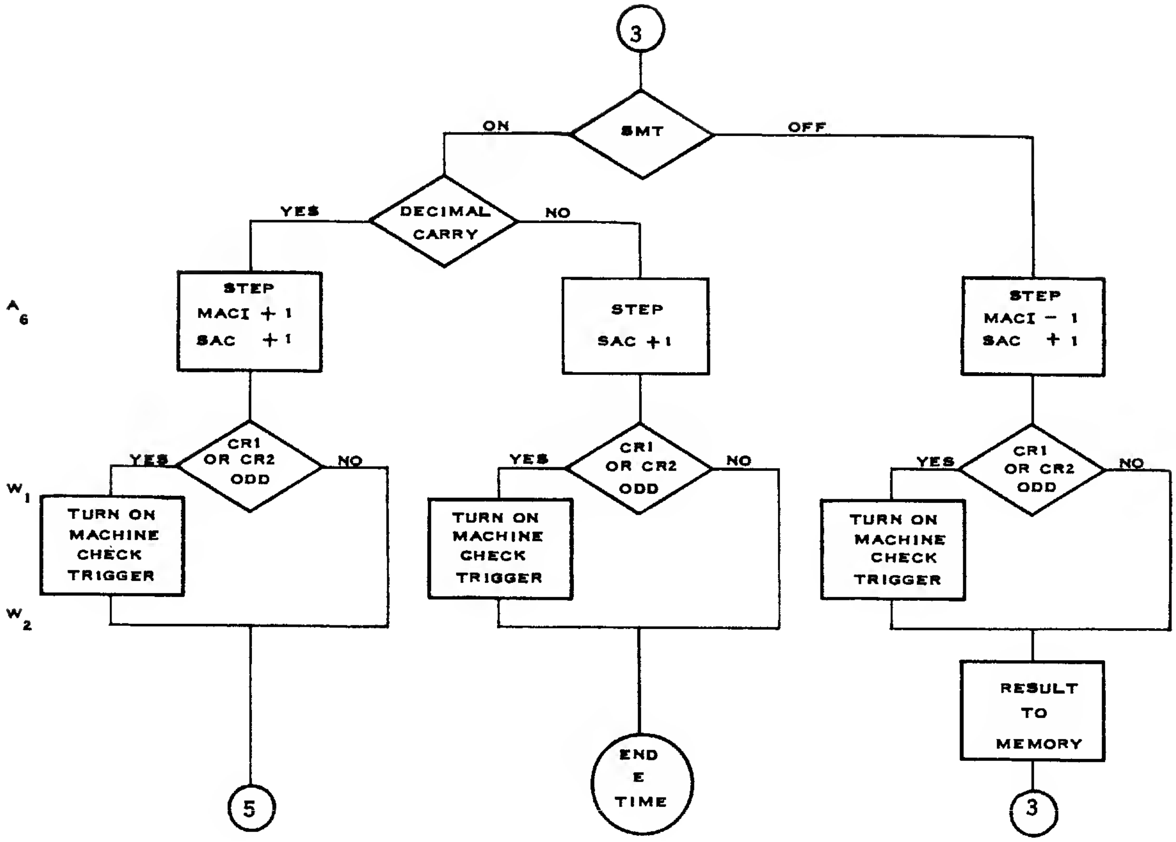


FIGURE 61

ADM

TYPE CYCLE IV
UNCONDITIONAL OPERATIONS:

READ MEMORY
CR1 TO TC
COMPLEMENT
DIGIT ADDER OUT
C-BIT GENERATOR OUT

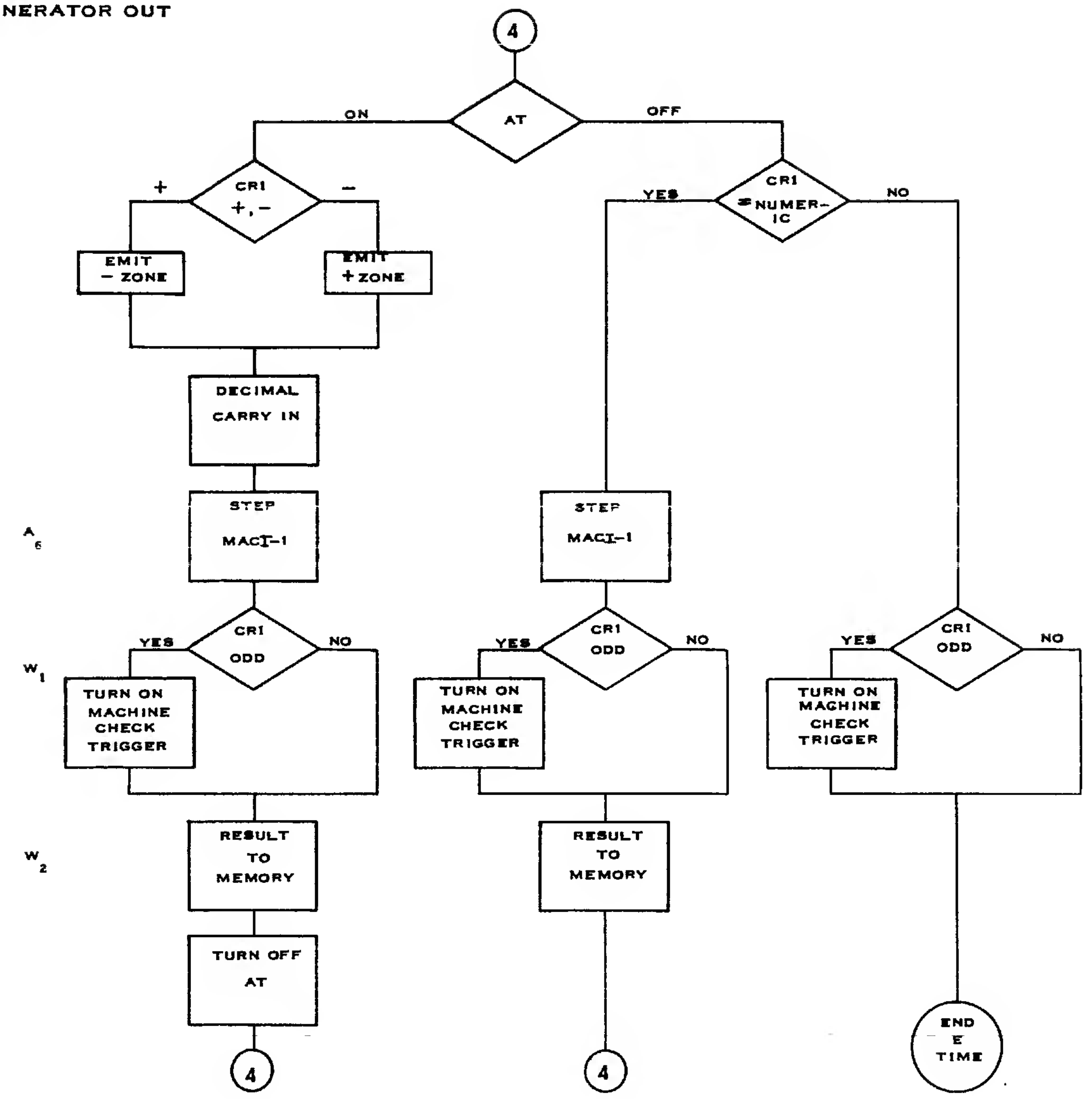


FIGURE 62

ADM

TYPE CYCLE V
UNCONDITIONAL OPERATIONS:

READ MEMORY
CR1 TO ADDER
DIGIT ADDER OUT
ZONE CARRY IN
ZONE ADDER OUT
C-BIT GENERATOR OUT
RESULT TO MEMORY

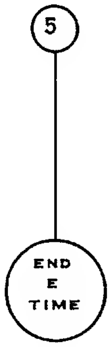


FIGURE 63

RD/WR/WRE

These three instructions (Figures 64 through 67) deal with the entry of information into, and exit of information from, the main frame of the 705 via peripheral equipment. Thus they are instructions which depend on components of the installation other than the 705 central processing unit - a fact that may cause the 705 to wait, or take idle cycles, until the relevant peripheral unit is ready.

As Figure 19 illustrates, when information is read into the 705, it enters through CR2. When information is written from the computer, it passes via CRl to the I/O (Input/Output) buses, which are the lines through which information passes between the 705 and its peripheral equipment. There are seven buses for reading and seven for writing.

In the RD instruction, only two functions are performed in the first cycle: a Prepare to Read Trigger in the input unit and a Read Call Trigger are turned on, then the I/O operation begins. The function of these triggers is as follows: The Prepare to Read Trigger alerts the input unit concerned. The Read Call Trigger tells the input unit to perform a reading operation.

The WR and WRE instructions are similar to the RD instruction. If all the following three conditions are met, the instruction is ended after the first execution cycle:

a. The WR or WRE instruction is addressed to a group mark in memory.

b. The SSR is equal to zero.

c. The Simultaneous Read/Write Trigger is in an Off condition.

In all other cases, the first cycle is followed by the I/O operation. Unless a group mark is addressed with SSR being equal to zero, the Write Call Trigger is always turned on, which tells the output unit to perform a writing operation. Whenever the simultaneous Read/Write Trigger is On, the Read Call Trigger is also activated.

In the I/O Operation, the 705 must wait until the peripheral unit selected is ready to send or receive a character. Basically, the rhythm of the reading in the I/O 1 operation depends on certain triggers which function as follows:

The Read Request Trigger tells the input unit selected that the 705 is ready to receive a character on its I/O buses. A Read Response is received from the input unit as the character is placed on the I/O buses. This Read Response turns on the Read Service Trigger which, when found in the On condition at W_2 time of an idle cycle, causes a Read Cycle to start. Writing is carried out in an equivalent manner.

If Read and Write Responses occur simultaneously, reading takes precedence over writing, as a character can be written at any time but can only be read when the selected input unit is ready. In a simultaneous reading and writing operation, the Read and Write Responses operate independently. Therefore, if the tape drives selected are not fully synchronized, it may be possible to read two characters before one is written or vice versa.

When the I/O 1 operation starts in a RD instruction, the 705 waits for the Read Response, while the megacycle clock continues operating. Thus, the 705 takes idle cycles, which have nine microseconds duration as the middle or arithmetic portion of the character cycle is omitted. The Read Response turns on the Read Service Trigger which causes the Read Request Trigger to be activated at the next W_2 time, followed by a Read Cycle. In the Read Cycle, the Read Service Trigger is turned off at R_0 time and the Read Request Trigger at W_2 time. Then, the 705 waits again for the next Read Response. When the end of a record is sensed in reading, then,instead of a Read Response, a Read Disconnect signal is returned to the 705. This turns off the Read Call Trigger, which has been On since the first type cycle, and ends the reading.

When writing is performed in the I/O 1 operation, a Write Response is returned to the 705 each time the output unit has completed writing a character, in order that the 705 place the next character on the I/O buses. The Write Request and Write Service Trigger operate in writing characters in a manner analogous to the use of the equivalent triggers in reading. The computer recognizes the end of the record to be written in the form of a group mark in memory, or 19999 (and also 39999 for a 705 Model II) in MAC I, during the Write Cycles of the I/O 1 operation and turns off the Write Call Trigger. After leaving the Write Cycle, the 705 recognizes the Read Disconnect Signal which signifies either that reading is ended or that no reading was done. When the 705 then receives a Write Disconnect signal from the output unit, it indicates that writing has been finished in the output unit and that any existing error conditions have been sensed and the I/O 1 operation is terminated.

When a simultaneous reading and writing operation is carried out, an On condition of the Read Service Trigger will prevent the Write Request Trigger from being turned on in order to ensure the processing in the Central Processing unit of only one character at a time.

Reading and writing operations are each ended by an I/O 2 cycle which is a single character cycle taken by the 705 to interrogate for error and End of File signals from the input/output units and to turn on the necessary indicator triggers. If the simultaneous Read/Write Trigger is On, both reading and writing must be ended before the I/O 2 cycle can begin.

The WRE instruction is performed in the same way as the WR instruction, except that a blank character is emitted and returned to memory instead of the character previously read out of memory. The WR (01) instruction differs from a WR (00) only in the limiting factor that terminates the writing process by turning off the Write Call Trigger. The RD (01) operation functions in the same way as the regular RD instruction, except that the characters read from the input unit, which in this case is normally a tape unit, are not placed in the memory of the 705.

If reading or writing is effected through a buffer unit, such as a 777 Tape Record Coordinator, the 705 does not have to take idle cycles between the Read or Write Cycles. The sampling of the Read or Write Response and Disconnect lines and Service Triggers is carried out simultaneously with the Read or Write Cycle, as the case may be. Also, the Read or Write Request Trigger remains in an On condition, but may be turned off during any of the successive W_2 times if the necessary Response has not been received. Therefore, in this case, the execution of the RD or WR instruction consists practically only of a succession of 9 microsecond Read or Write Cycles.

RD

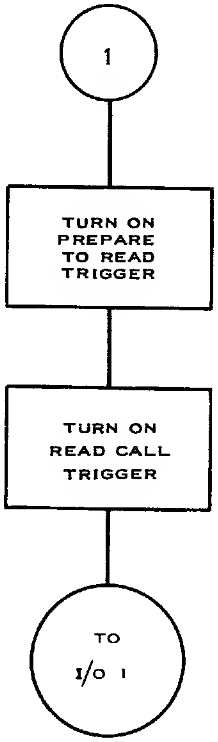


FIGURE 64

WR/WRE

UNCONDITIONAL OPERATION:

READ MEMORY

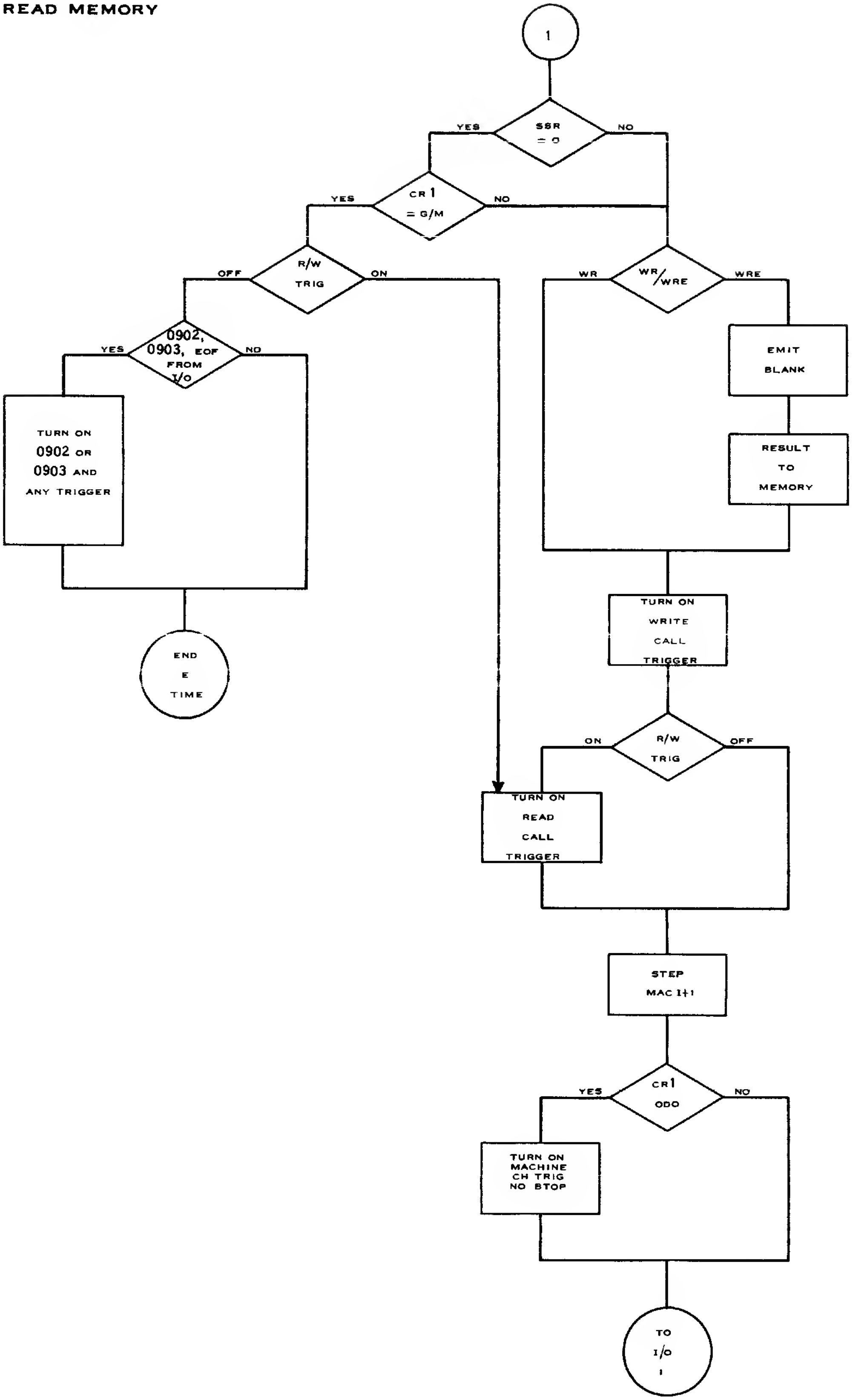


FIGURE 65

I/O - W CYCLE

UNCONDITIONAL OPERATIONS:
CR2 TO TC
SUPPRESS ADDER CARRY

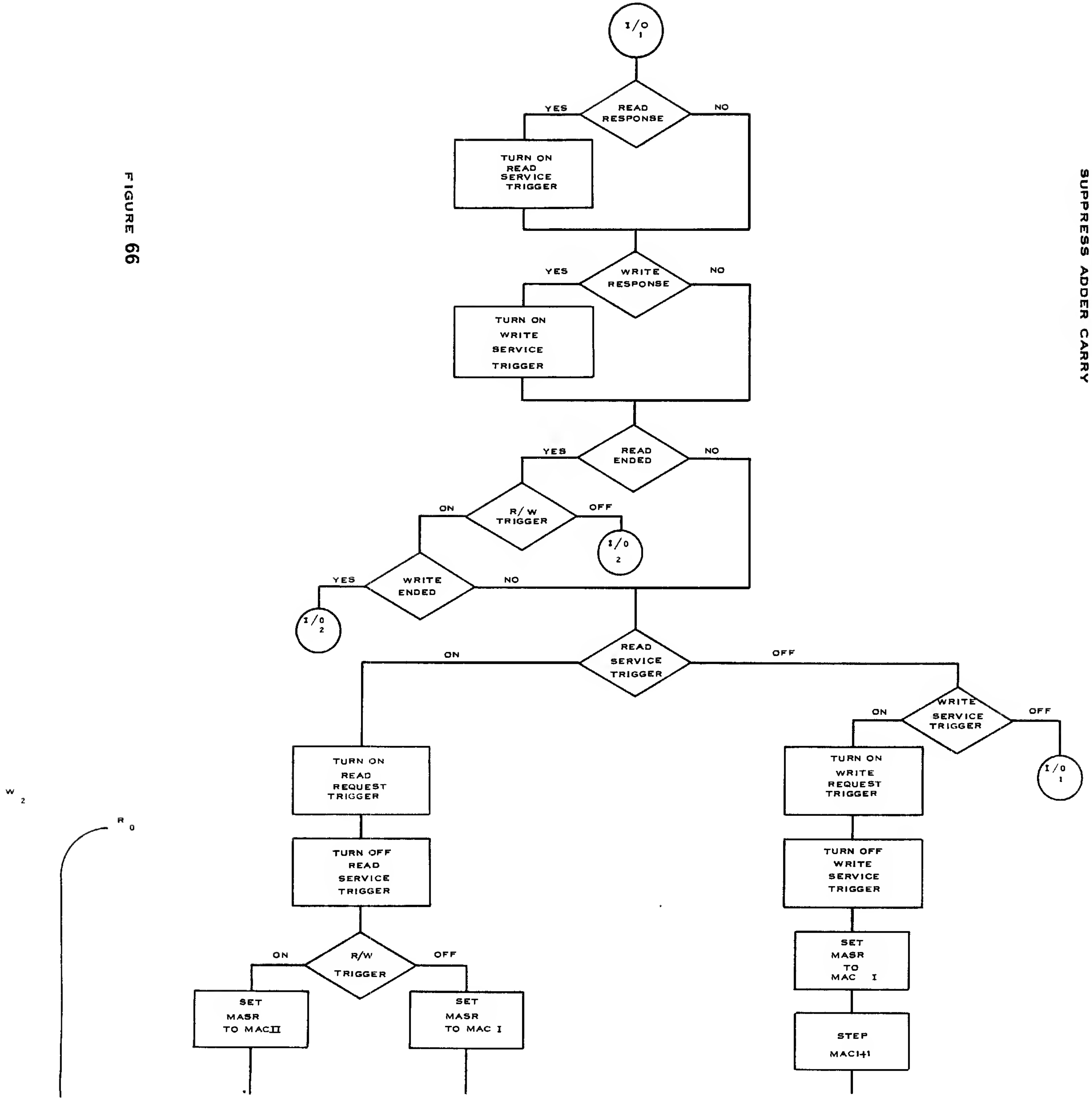
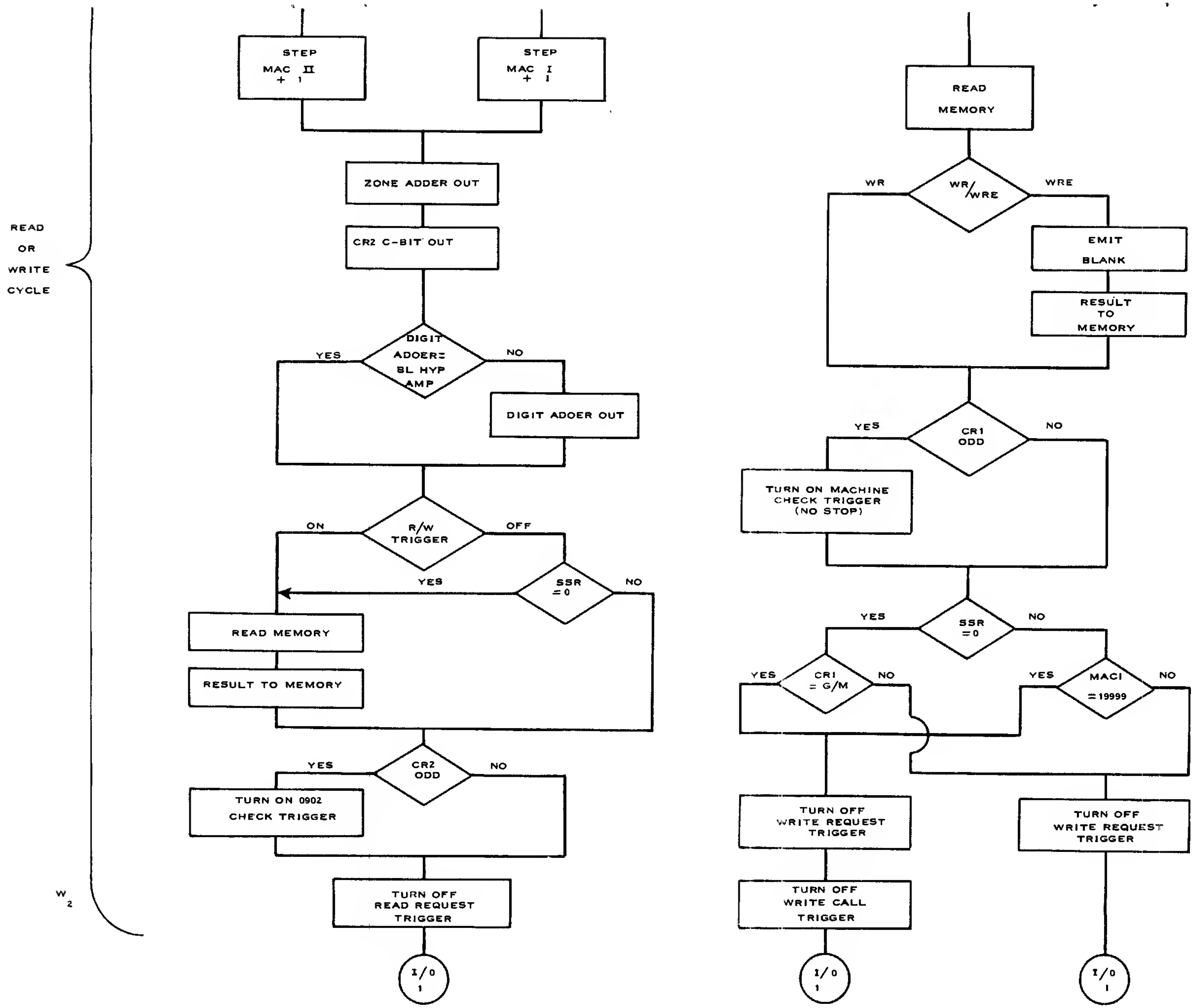


FIGURE 66


READ OR WRITE CYCLE
W 2
STEP MAC II + 1
STEP MAC I + 1
ZONE ADDER OUT
CR2 C-BIT OUT
DIGIT ADDER= BL HYP AMP
YES
NO
DIGIT ADDER OUT
R/W TRIGGER
ON
OFF
SSR =0
YES
NO
READ MEMORY
RESULT TO MEMORY
CR2 ODD
YES
NO
TURN ON 0902 CHECK TRIGGER
TURN OFF READ REQUEST TRIGGER
I/O 1
READ MEMORY
WR/WRE
WR
WRE
EMIT BLANK
RESULT TO MEMORY
CR1 ODD
YES
NO
TURN ON MACHINE CHECK TRIGGER (NO STOP)
SSR =0
YES
NO
CR1 = G/M
YES
NO
MACI =19999
YES
NO
TURN OFF WRITE REQUEST TRIGGER
TURN OFF WRITE CALL TRIGGER
I/O 1
TURN OFF WRITE REQUEST TRIGGER
I/O 1

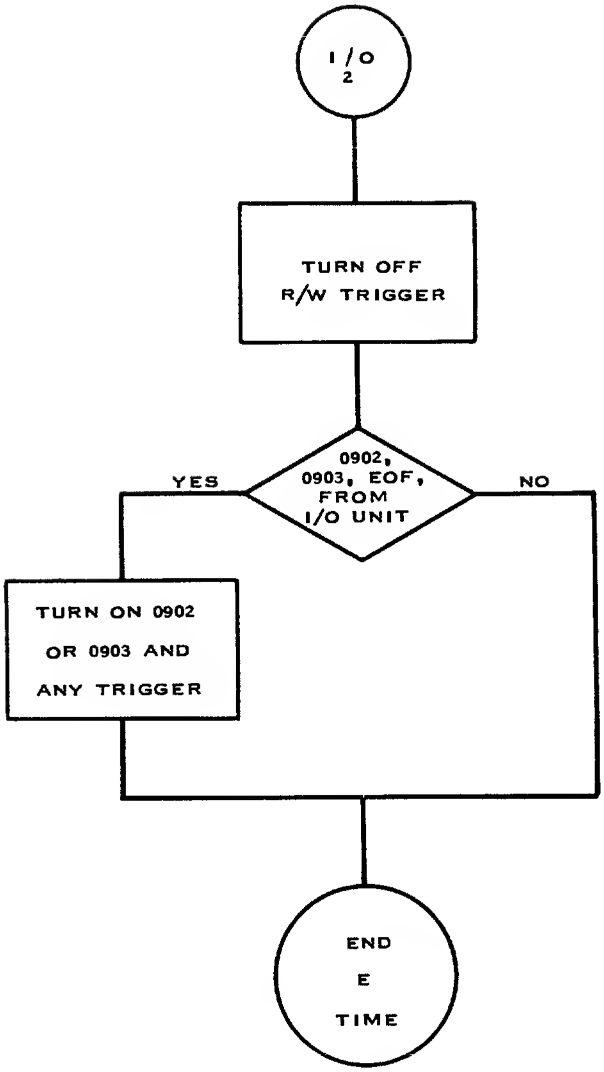

I/O
2
TURN OFF
R/W TRIGGER
0902,
0903, EOF,
FROM
I/O UNIT
YES
NO
TURN ON 0902
OR 0903 AND
ANY TRIGGER
END
E
TIME

FIGURE 67

CTRL

The various CTRL instructions (Figure 68) make use of the Auxiliary Trigger (AT) to cause the 705 to perform the last ECC which ends E-time. As in the RD, WR, and WRE instructions, the 705 depends on peripheral equipment to carry out its part of the instruction function. The control function circuitry once activated remains so until the input/output unit has completed the indicated operation or until it releases the central processing unit, depending on the nature of the CTRL instruction. At that time, a Disconnect signal is received from the peripheral equipment, turning on the AT.

At W_2 time of each ECC, the AT is interrogated and until it is found to have been turned on, the computer continues taking idle cycles.

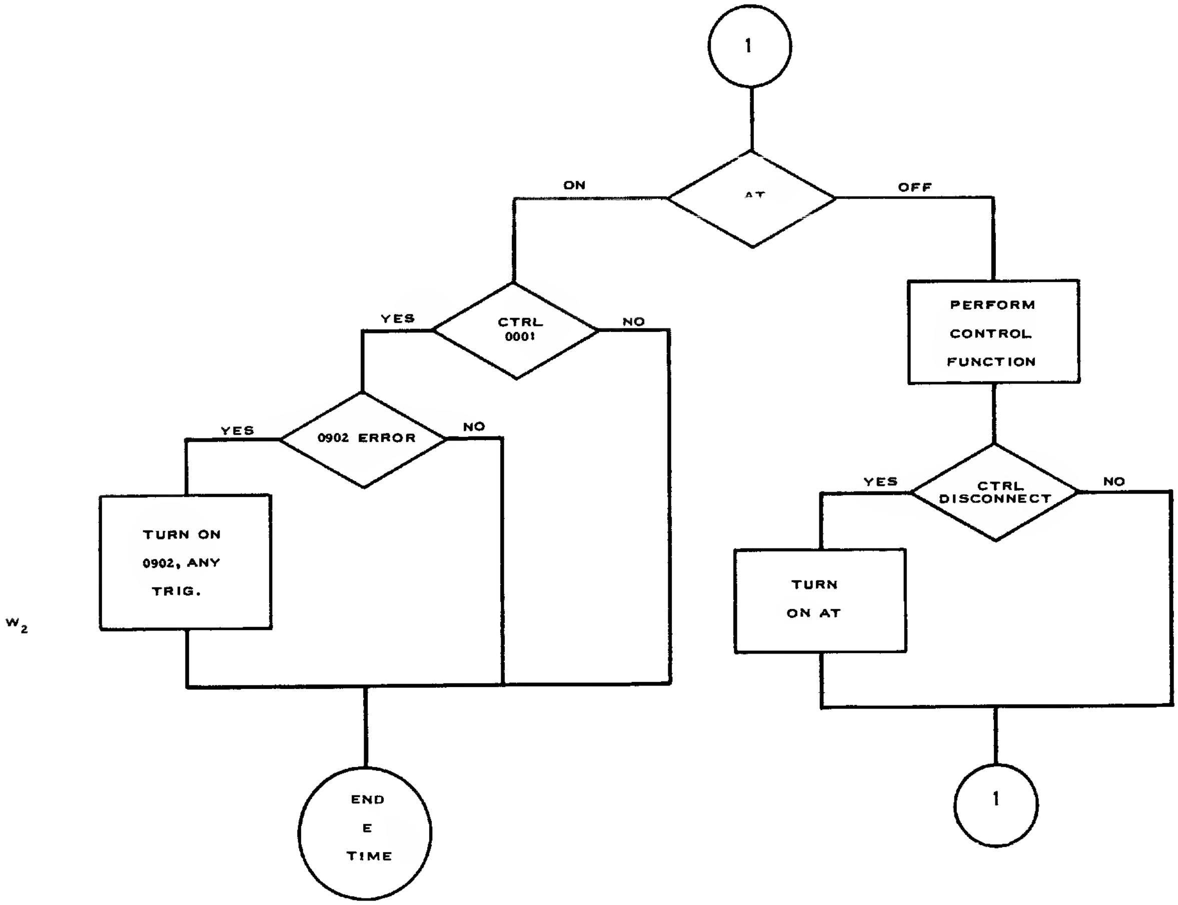


FIGURE 68

MPY

The MPY instruction (Figures 69 through 74) employs five type cycles to perform multiplication. Basically, multiplication is carried out in the following pattern:

The units digit of the multiplier, which is the number in storage, goes to the multiply register; then it multiplies first the units digit of the multiplicand (the number in the memory field addressed) and, thereafter, in succession from right to left, the other digits in the multiplicand field. Subsequently, the tens digit of the multiplier goes to the multiply register and the operation is repeated, with the partial product previously obtained being added to the new product being developed. In this fashion, multiplication proceeds until the multiplier is exhausted which is indicated by the storage mark being sensed in CR2.

The overall flow of this multiplication operation in the five type cycles is as follows:

In Type Cycle I, the current multiplier digit is placed in the multiply register. If a storage mark is sensed, the 705 will progress directly to Type Cycle V which is used to end the multiply operation. In all other cases, Type Cycle II is entered in which the current multiplier digit multiplies the units position of the multiplicand. Type Cycle III, beginning unconditionally thereafter, continues the multiplication by the same multiplier digit. The 705 repeats Type Cycle III until all remaining digits in the multiplicand have been multiplied, progressing from right to left, at the same time adding to the current product any partial product which may have been previously developed in storage. When the end of the memory field is recognized as a non-numeric in CR1, Type Cycle IV starts, during which a storage mark is placed at the left end of the partial product field and the AT is turned on to indicate the existence of a partial product. From Type Cycle IV, the computer returns to Type Cycle I to move the next multiplier digit into the multiply register. This loop of Type Cycles I-IV is continued until all multiplier digits have been depleted. In Type Cycle V the Starting Point Counter (SPC) is stepped to the unit digits of the product field.

In order to explain the detailed events and decisions occurring in the individual Type Cycles, the execution of a simple example of multiplication will be traced through the cycles. Let the memory field addressed by the MPY instruction be b2$\overset{+}{4}$ and let the multiplier in storage be @37 with the storage sign set to plus. For the purposes of this example, accumulator positions will be assumed to be numbered from 1 - 256 and SPC to be at accumulator position 200. During ECC1 in Type Cycle I, the digit 7 is read from storage and passed through CR2 into the multiply register. A plus zone is placed over the 7 when it is returned to storage, in order that the units position of the multiplier field can be recognized in Type Cycle V. The Storage Address Counter (SAC) is stepped to accumulator position 72. (As always, SAC was set to SPC in I-time; it was also explained in the chapter on Storage Address Control that SAC can be stepped 128 positions in one machine cycle.)

In Type Cycle II the units position of the multiplicand $\overset{+}{4}$ is read from memory, multiplied by 7, and the units position of the result, 8, is placed in position 72 of the accumulator. The character that was previously in that accumulator position is read into CR2 and lost since CR2 is not routed out. During the first passing of the loop of Type Cycles I-IV, the AT is in an Off condition, indicating that no partial product has been obtained previously and that thus nothing is to be added to the product currently being developed. (The AT is turned off in every I-time and therefore is in an Off condition during E-time unless specifically turned on during the execution of an instruction.) SAC is stepped to accumulator position 73 in order to be ready to place the next product digit in the product field in storage, and SPC is stepped to position 201, where the next multiplier digit is located. MAC 1 is stepped down one position so that the multiplicand digit 2 can be read out of memory during the next cycle.

Multiplication of two individual digits in the 705 is achieved in a way very similar to the process of addition discussed previously. Whenever a digit from CR1 is moved to the digit adder through the multiply circuits, multiplication by 1 is assumed and the digit passes unchanged. If some digit, other than 1 is in the multiply register, it will indicate a fixed path of logical circuitry, such that the resulting digit sent to the adder is the units digit of the product with the proper carry being retained in the multiply circuitry.

Type Cycle III resembles closely the functions of Type Cycle II: The multiplier digit in the multiply register multiplies successive multiplicand digits until the end of the memory field is sensed. These two Type Cycles must, however, be separate cycles, for in Type Cycle II a signed multiplicand digit must be found as otherwise a sign check will result. In Type Cycle III, however, recognition of a non-numeric signifies the end of the multiplicand field and therefore terminates further executions of Type Cycle III.

In Type Cycle III, the multiplicand digit 2 in memory is multiplied by the 7 in the multiply register, giving a result of 14. But a 2 had been held in the multiply carry circuits from Type Cycle II (7 x 4 = 28) and therefore a 6 is placed in accumulator position 73. Only SAC and MAC I are stepped in Type Cycle III, leaving SPC at the accumulator position of the next multiplier digit. Type Cycle III is repeated, and the carry of 1 left in the multiply carry circuits is placed in accumulator position 74. If there had been no carry, a zero would have been placed in this position because the digit adder is routed out. During this second passing of Type Cycle III, a non-numeric (blank) was read and Type Cycle IV is accordingly entered.

In Type Cycle IV, SAC is set to SPC which is still standing at accumulator position 201, and the AT is turned on to indicate that a partial product exists. Also, SMT is turned off in case it had been turned on in any preceding type cycle, as it, if sensed in an On condition in Type Cycle I which follows thereafter, would cause a false end of multiplication. A storage mark is also entered at the left of the partial product, in this case position 75 of the accumulator.

In Type Cycle I which this time does not coincide with ECC1, the multiplier tens digit, 3, at accumulator position 201 will set the multiply register. SAC is again stepped 128 and will therefore be at accumulator position 73 (the tens position of the final product). In Type Cycles II and III, 3 will now multiply the multiplicand. In the same cycle that a digit is multiplied, it is also added to the digit in the product field in storage indicated by SAC. As SPC is stepped up one accumulator position for each new multiplier digit, and as SAC is stepped in exactly the same way, 128 positions from SPC, it follows that each successive product of one multiplier digit with the whole multiplicand is added one position to the left of the preceding accumulated partial product.

At the end of Type Cycle IV, after the loop of Type Cycles I-IV has been completed for the second time, the product field in accumulator position 76-72 is @0888. When Type Cycle I begins for the third time, the storage mark in accumulator position 202 will be read, and SPC and SAC will be stepped back to accumulator position 201, the left end of the multiplier field.

Then Type Cycle V begins in which SAC and SPC are stepped down until the right end of the multiplier field, recognized as being a non-numeric, is found. It may be remembered here that in the detailed description of Type Cycle I, it was explained that the units digit of the multiplier was zoned in order to indicate the right-hand end of the multiplier field. When this non-numeric is found, SPC and SAC will both be set again to accumulator position 200. Then SPC is stepped 128, putting it at accumulator position 72, the right-hand end of the product field, and E-time is ended.

Multiplication can be accomplished with alphabetic characters in storage, but all zones are removed as the zone adder is never routed out.

MPY

OVERALL FLOW

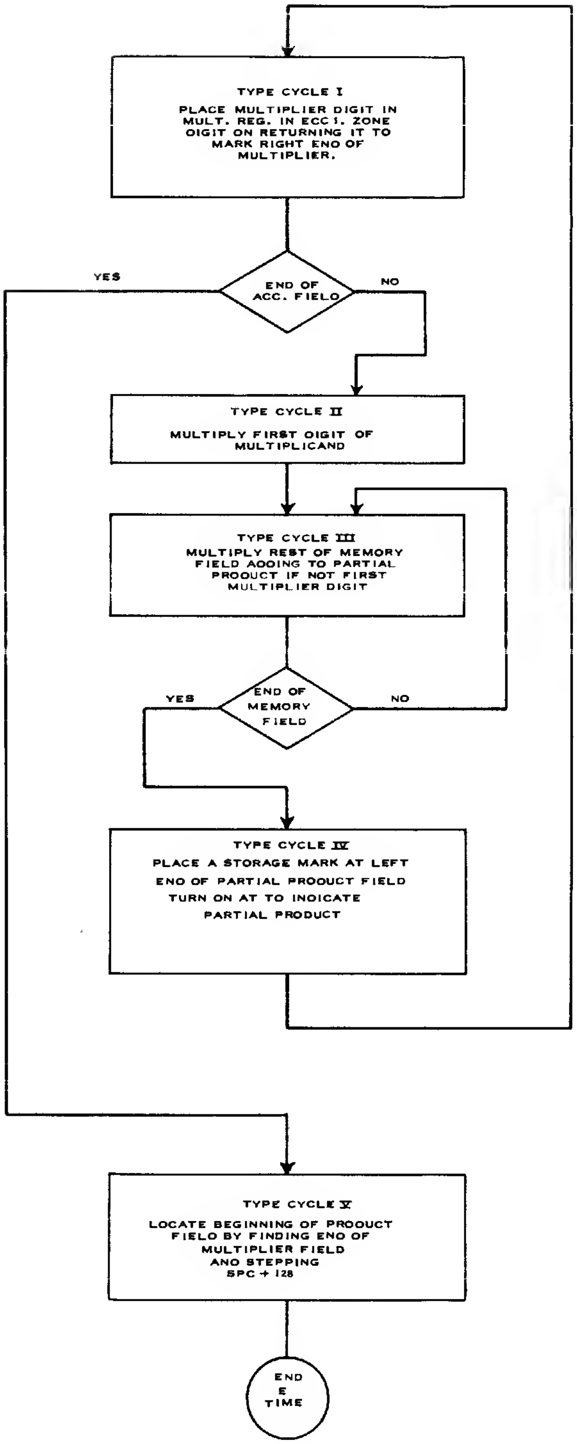

TYPE CYCLE I
PLACE MULTIPLIER DIGIT IN MULT. REG. IN ECC 1. ZONE OIGIT ON RETURNING IT TO MARK RIGHT ENO OF MULTIPLIER.
END OF ACC. FIELO
YES
NO
TYPE CYCLE II
MULTIPLY FIRST OIGIT OF MULTIPLICANO
TYPE CYCLE III
MULTIPLY REST OF MEMORY FIELD AOOING TO PARTIAL PROOUCT IF NOT FIRST MULTIPLIER DIGIT
END OF MEMORY FIELD
YES
NO
TYPE CYCLE IV
PLACE A STORAGE MARK AT LEFT ENO OF PARTIAL PROOUCT FIELD
TURN ON AT TO INOICATE PARTIAL PRODUCT
TYPE CYCLE V
LOCATE BEGINNING OF PROOUCT FIELO BY FINDING ENO OF MULTIPLIER FIELD ANO STEPPING SPC + 128
END E TIME

FIGURE 69

MPY

TYPE CYCLE I
UNCONDITIONAL OPERATIONS:

READ STORAGE
CR2 TO T C
DIGIT ADDER OUT
C-BIT GENERATOR OUT

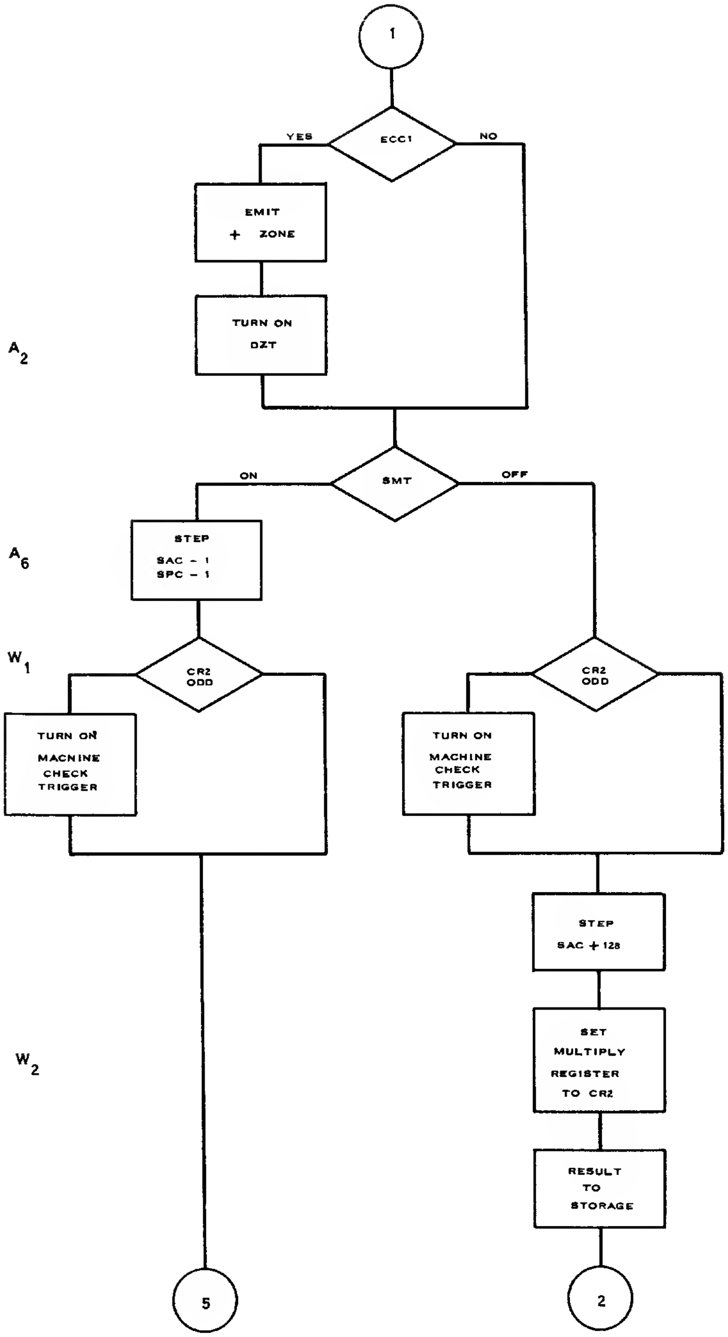


FIGURE 70

MPY

TYPE CYCLE II
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CR1 TO ADDER
DIGIT ADDER OUT
C-BIT GENERATOR OUT
RESULT TO STORAGE

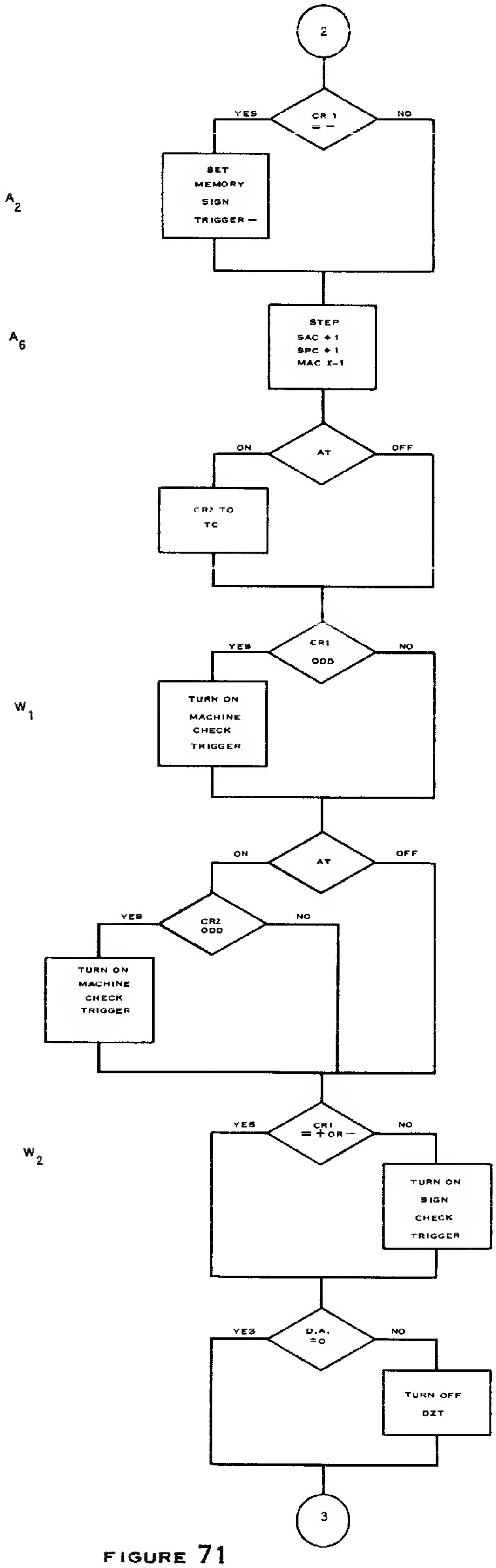

2
YES
CR 1 = –
NO
A_2
SET MEMORY SIGN TRIGGER –
A_6
STEP SAC + 1 SPC + 1 MAC I-1
ON
AT
OFF
CR2 TO TC
YES
CR1 ODD
NO
W_1
TURN ON MACHINE CHECK TRIGGER
ON
AT
OFF
YES
CR2 ODD
NO
TURN ON MACHINE CHECK TRIGGER
YES
CR1 = + OR –
NO
W_2
TURN ON SIGN CHECK TRIGGER
YES
D.A. = O
NO
TURN OFF DZT
3

FIGURE 71

MPY

TYPE CYCLE III
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
DIGIT ADDER OUT
C-BIT GENERATOR OUT
RESULT TO STORAGE

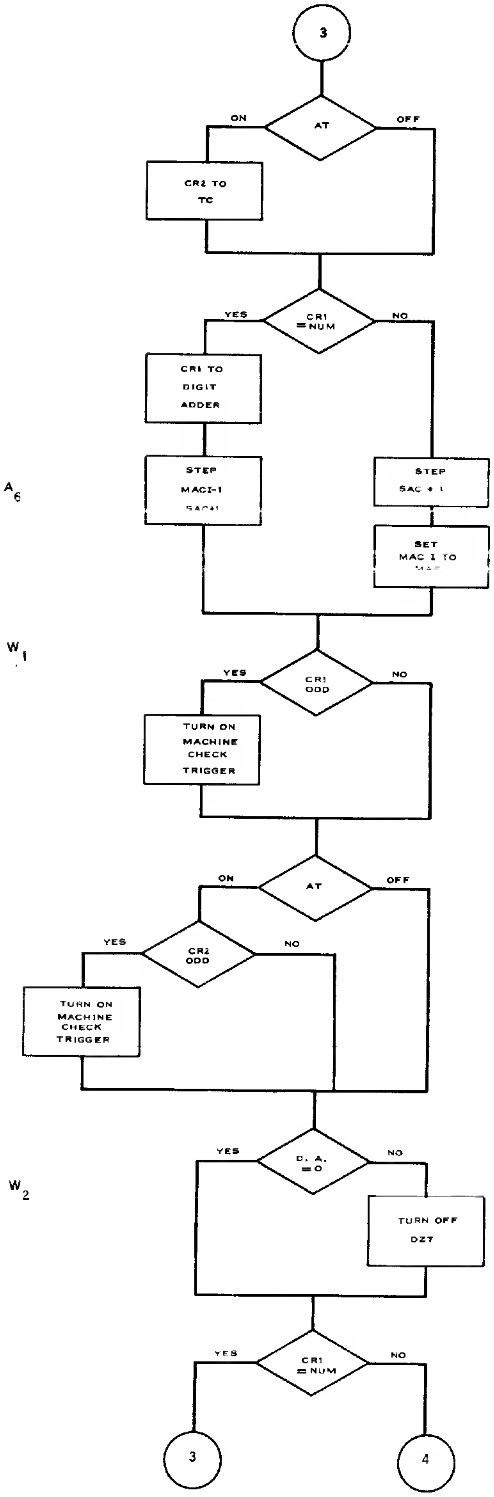


FIGURE 72

MPY

TYPE CYCLE IV
UNCONDITIONAL OPERATIONS:

READ STORAGE
RESULT TO STORAGE

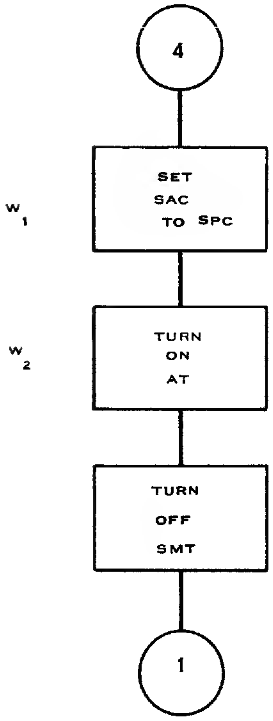


FIGURE 73

MPY

TYPE CYCLE V
UNCONDITIONAL OPERATIONS:

READ STORAGE
CR2 TO TC

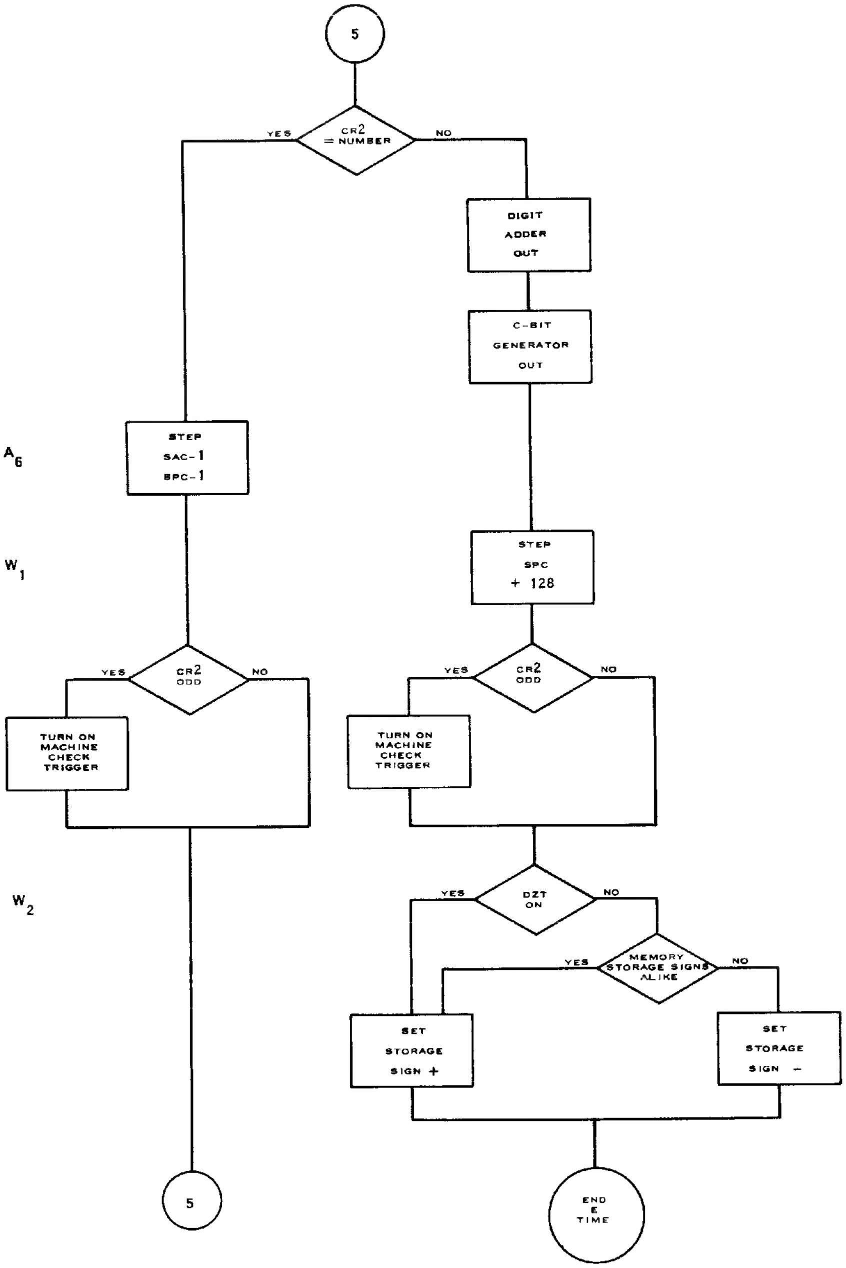


FIGURE 74

DIV

The DIV instruction (Figures 75 through 86) has ten types of cycles. The first four type cycles are housekeeping type cycles which, provided there is no overflow condition, are used only once in the execution of any one DIV instruction. Type Cycles V-IX carry out the actual division, and Type Cycle X places SPC at the units position of the quotient field.

In the 705, division is accomplished by repeated subtraction of the divisor from the dividend. At first, the divisor, which shall be assumed to consist of n digits, is subtracted from the n+1 high-order digit of the dividend. To develop the quotient, a count is maintained of the number of these subtractions, which are repeated until a negative or zero remainder results. In the case of a negative remainder, the divisor is added back to yield a positive remainder which is extended to the right by the inclusion of the (n+2) th digit of the dividend, counting from the left. Thereafter, repeated subtraction of the divisor from this extended remainder follows, with the count of this set of subtractions stored to the right of the high-order quotient digit previously obtained. This process continues from left to right until the dividend is exhausted. The quotient is not rounded and any remainder, unless deliberately recovered by programming, is lost. If the Absolute Value Rule (i. e. the rule requiring that the absolute value of the divisor be greater than the absolute value of an equal number of the high-order digits of the dividend) has not been followed, the number of subtractions, when the divisor is subtracted from the n+1 high-order positions of the dividend, will exceed 9 and an overflow check stop will result. Similarly, if the Length Rule (i. e., the rule requiring that the number of digits in the dividend be greater than the number of digits in the divisor) is violated, it is clear from the above description that division cannot be accomplished.

In Type Cycle I, a storage mark is placed to the right of the units position of the dividend. This storage mark is used later to indicate the end of the dividing process.

Type Cycle II steps up SPC and SAC one position at a time until the storage mark is reached which adjoins the high-order position of the dividend. Then SPC is stepped back to the high-order position of the accumulator and SAC is stepped +128.

In Type Cycle III, a storage mark is deposited in the accumulator position to which SAC was stepped previously, in order to mark the left-hand (high-order) end of the quotient. SAC is then set to SPC.

In Type Cycle IV, SAC and SPC which are standing at the high-order position of the dividend are stepped down (to the right) one at a time while simultaneously

MAC I is stepped down (to the left) one at a time from the units position of the divisor at which it has been standing. This stepping continues until a non-numerical character is recognized in CR1; in other words, when the character limiting the divisor field has been reached. In this type cycle, it is important to differentiate between the units position of the divisor encountered first, where absence of both a plus and a minus sign results in a sign check, and between the subsequent positions in the memory field where the presence of any zoning indicates the end of the divisor field. This differentiation is accomplished by using the AT. It should be noted that, if the length of the divisor be n characters, the position to which SAC and SPC will have stepped is the (n+1) th position of the dividend counting from the left as MAC I has stepped one beyond the high-order position of the divisor. During the last cycle of Type Cycle IV, MAC I is set back to MAR which, of course, stands at the units position of the divisor. If during this type cycle in any character cycle, a storage mark is sensed, it will indicate that the dividend has the same as, or a smaller number of digits than, the divisor. This condition, which violates the Length Rule, will terminate division in this type cycle.

In Type Cycle V, the storage field starting with the character to which SAC had been set at the end of Type Cycle IV is complemented, and in order to achieve a 10's complement, a decimal carry is routed in during the first ECC of this type cycle. The memory field, the divisor, is subtracted from the left n+1 positions of the dividend by being added to the 10's complement of this part of storage. When the non-numeric at the high-order end of the memory field is reached, the subtraction is completed. During this type cycle the 705 avails itself of the AT to differentiate between the first ECC in which the decimal carry is routed in unconditionally and the subsequent ECC's in which this is not done. During the last character cycle in this type cycle after the non-numeric has been recognized in CR1, SAC is stepped +128 to the high-order quotient position, MAC I is set to MAR, and the AT is turned on.

Then Type Cycle VI starts. If there is a decimal carry-in at the first character cycle of this type cycle, it signifies that there is either no remainder or that the last subtraction (in Type Cycle V or VII), in effect, resulted in a minus remainder. A "1" is set in CR1 and deposited at the storage position indicated by SAC which, in view of its stepping in the prior type cycle, is at the quotient digit currently being developed. This process is the counting of subtractions which creates the quotient as explained previously. Subtractions subsequent to the first one are carried out in Type Cycle VII, but the counting is always done in Type Cycle VI. Therefore, it is necessary to differentiate in this type cycle between the first count when the character previously in the quotient position has to be eliminated, and subsequent counts when the digit in the quotient is a partial count to which the new "1" count has to be added. The 705 makes this differentiation by means of the AT: if the AT is in an "ON" condition, the character in storage indicated by SAC is not passed through CR2 and the adder circuits, and thus eliminated.

Type Cycle Vll carries out all subtractions subsequent to the first. lt differs from Type Cycle V basically only in the fact that the accumulator field operated on is not again complemented. As in Type Cycle V, the AT is used to differentiate between the first ECC during this type cycle and subsequent ones. As indicated, after completion of the subtraction in Type Cycle V, the 705 enters again Type Cycle VI where it is established whether the subtraction decreased the remainder to zero or whether an over-subtraction resulted.

lf in Type Cycle Vl, either in the case of a positive or zero remainder, the addition to the quotient digit being developed results in a carry (and this, of course, can only happen in the first series of subtractions when the high-order quotient digit is being developed), an overflow condition exists. In other words, the Absolute Value Rule has been violated. In case of the overflow condition, the 705 enters again Type Cycle ll and, as the AT was turned on in Type Cycle VI, ends execution time in Type Cycle lll.

This process of repetitive subtractions and additions of the count to the quotient digit continues until a zero remainder or negative remainder results. In the case of a negative remainder, Type Cycle VIII is entered, but in the case of a zero remainder Type Cycle IX follows directly.

In Type Cycle VIll, the storage field is recomplemented and the last subtraction which brought about a minus remainder is reversed. The AT is again used to differentiate between the units position of the divisor which is signed and the non-numeric which limits the divisor field. This non-numeric, when recognized in CRl, causes Type Cycle IX to commence.

In Type Cycle lX, SAC and SPC are stepped down by one, the Remainder Trigger (RT) is turned off because the prior set of subtractions is ended, and the AT is turned on. Type Cycle V follows.

In Type Cycle V, there are then two basic possibilities:

1. When the character to which SAC had stepped is recognized as a character other than a storage mark, the first out of a set of repetitive subtractions to develop the next lower quotient digit is started.

2. If the character indicated by SAC is, however, a storage mark, then the process of division is ended and SPC is stepped up to the units position of the dividend whereupon Type Cycle IX is re-started.

Previously, when Type Cycle IX was entered from Type Cycle VII or VIII, the AT was in an "Off" condition. Now, however, it is still "On" having been turned on when Type Cycle IX had been passed previously.

In Type Cycle IX, MAC I, which had stood at the units position of the divisor, is stepped to the 10's position, and SPC is stepped +128 positions to the right of the quotient. It should be noted here when referring to previous type cycles, that the high-order position of the quotient is 128 positions to the left of the high-order position of the dividend. Also, the quotient resulting from 705 division will always consist of as many digits as the number of digits of the dividend less the number of digits of the divisor. In order to let SPC stand, therefore, at the units position of the quotient upon the end of the DIV instruction, it is still necessary to step up SPC by as many storage positions as the divisor has digits. This is accomplished in Type Cycle X which examines the divisor field starting with the 10's position and which is completed upon sensing the non-numerical character terminating the divisor field. During the last ECC of Type Cycle X, the sign of the accumulator is adjusted to conform with the rule of signs of division.

In Figure 86 a division example has been worked out in full. The division of 186 by 9 is carried out showing the status of the dividend and quotient fields in the accumulator and of SPC and SAC at the end of I-time and at the end of each type cycle, each time it is passed in the sequence of execution. It may be helpful to follow the above explanation of what each type cycle accomplishes in the light of the example in Figure 86.

DIV

OVERALL FLOW

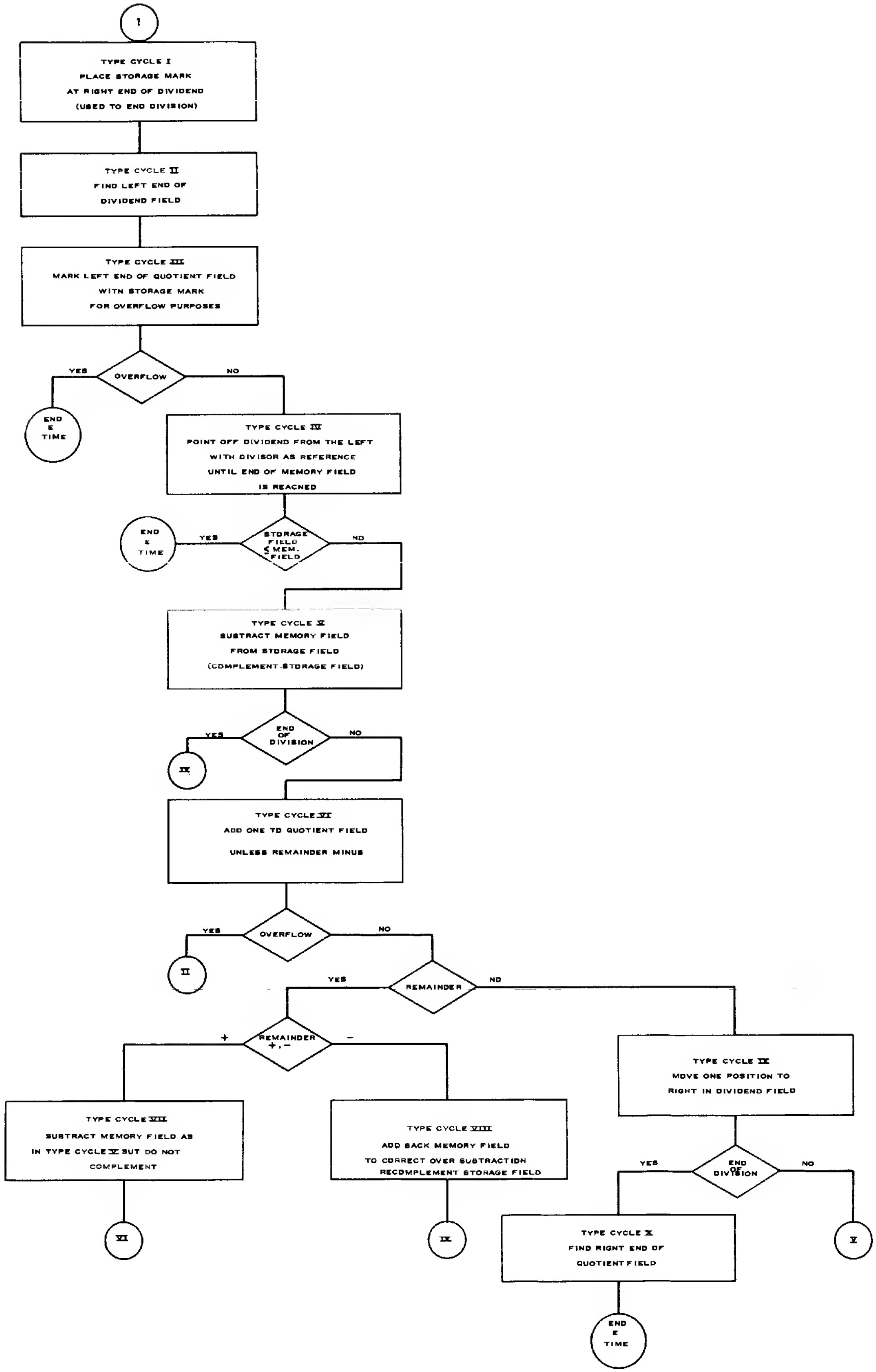


FIGURE 75

DIV

TYPE CYCLE I

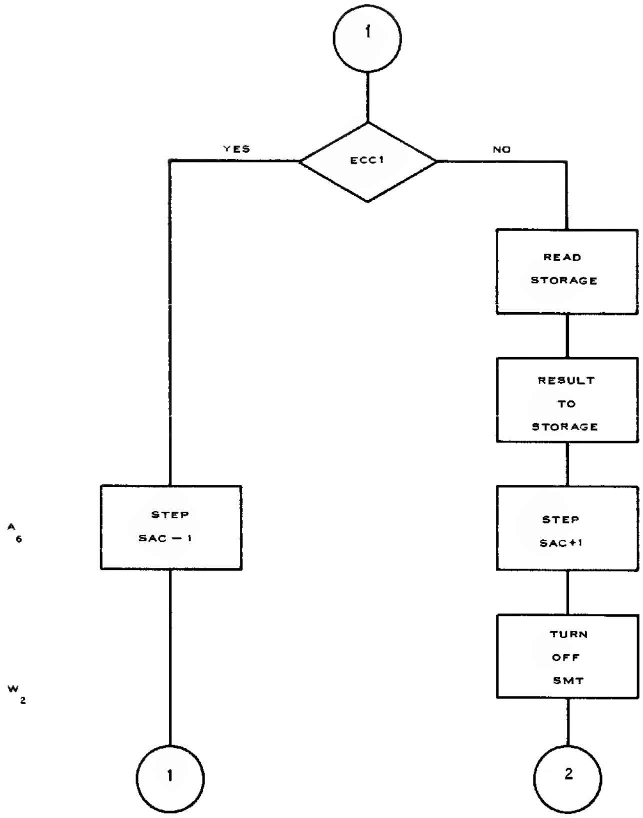


FIGURE 76

DIV

TYPE CYCLE II
UNCONDITIONAL OPERATION:

READ STORAGE

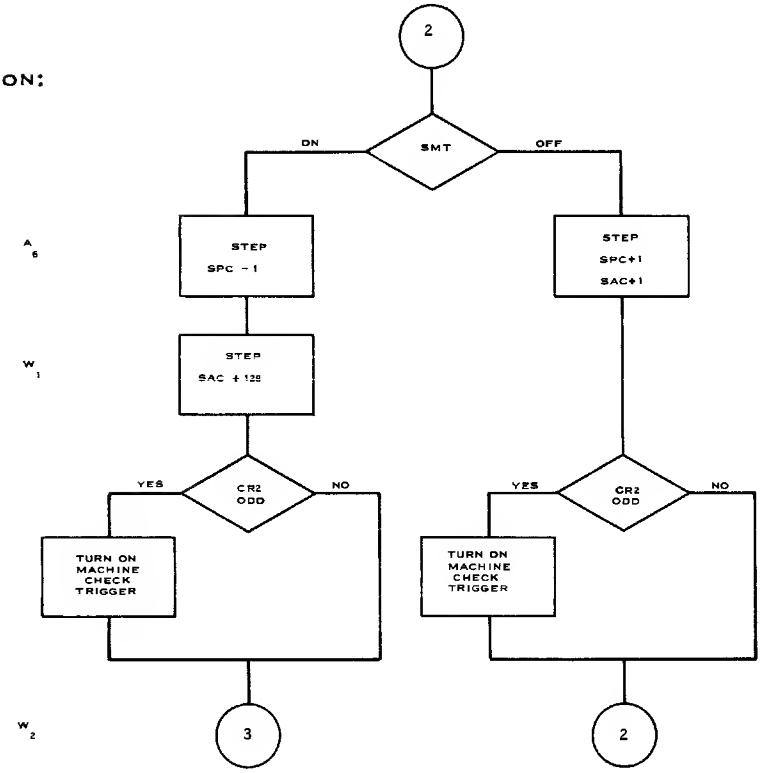


FIGURE 77

DIV

TYPE CYCLE III

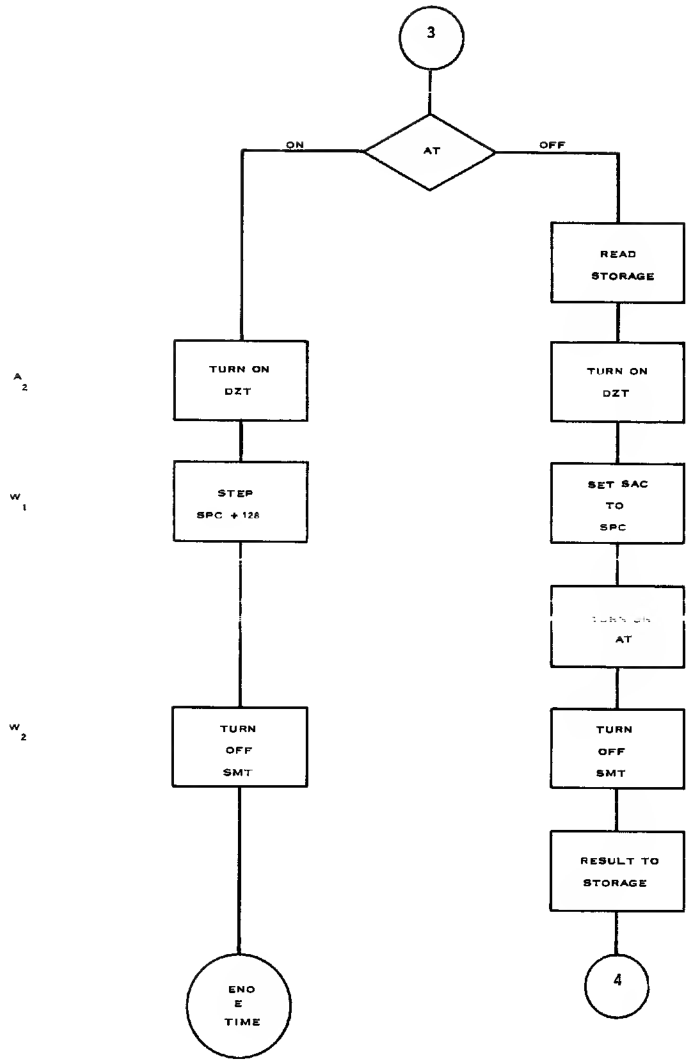


FIGURE 78

DIV

TYPE CYCLE IV
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE

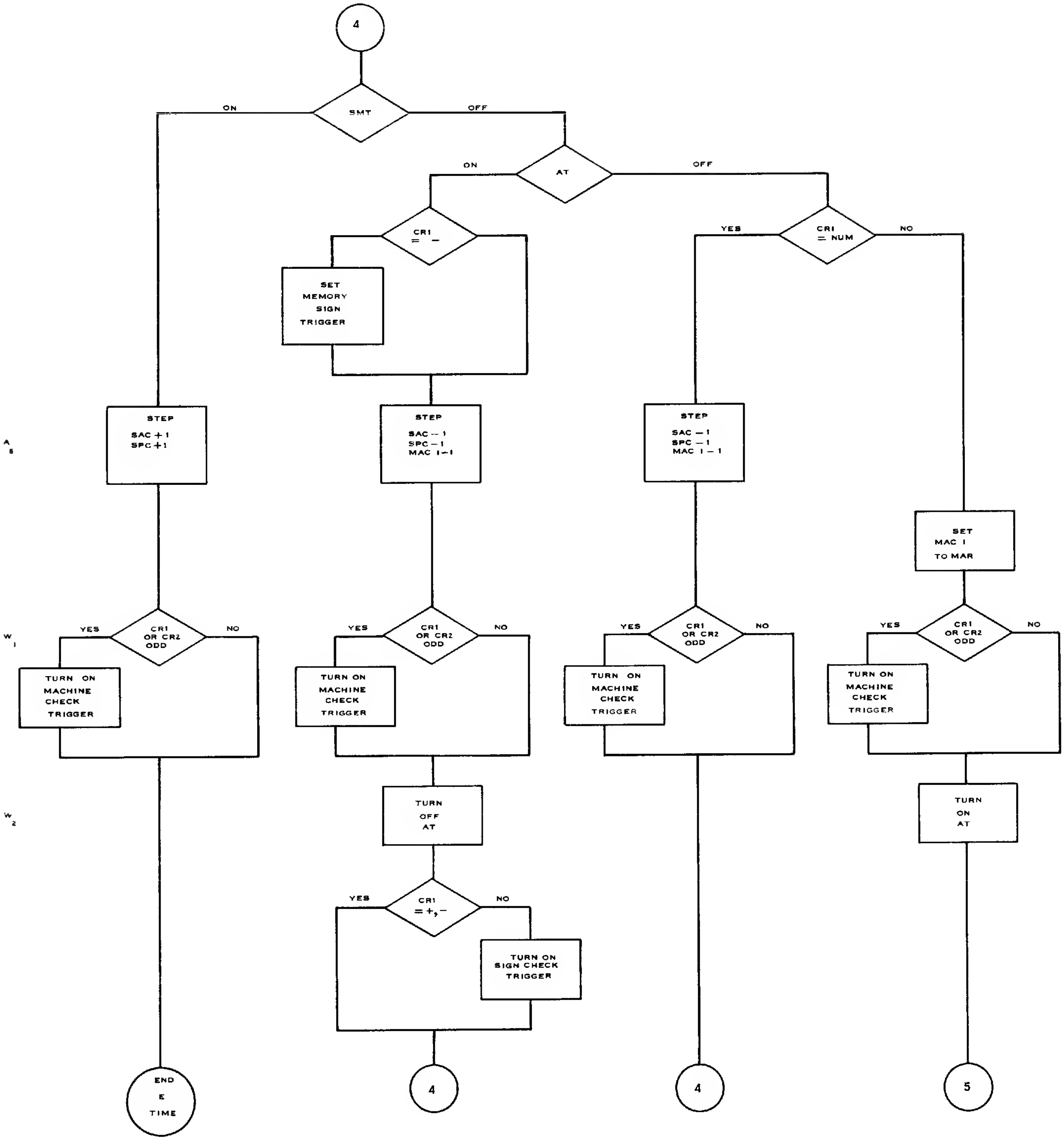


FIGURE 79

DIV

TYPE CYCLE V
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CR2 TO TC
COMPLEMENT
DIGIT ADDER OUT
C—BIT GENERATOR OUT

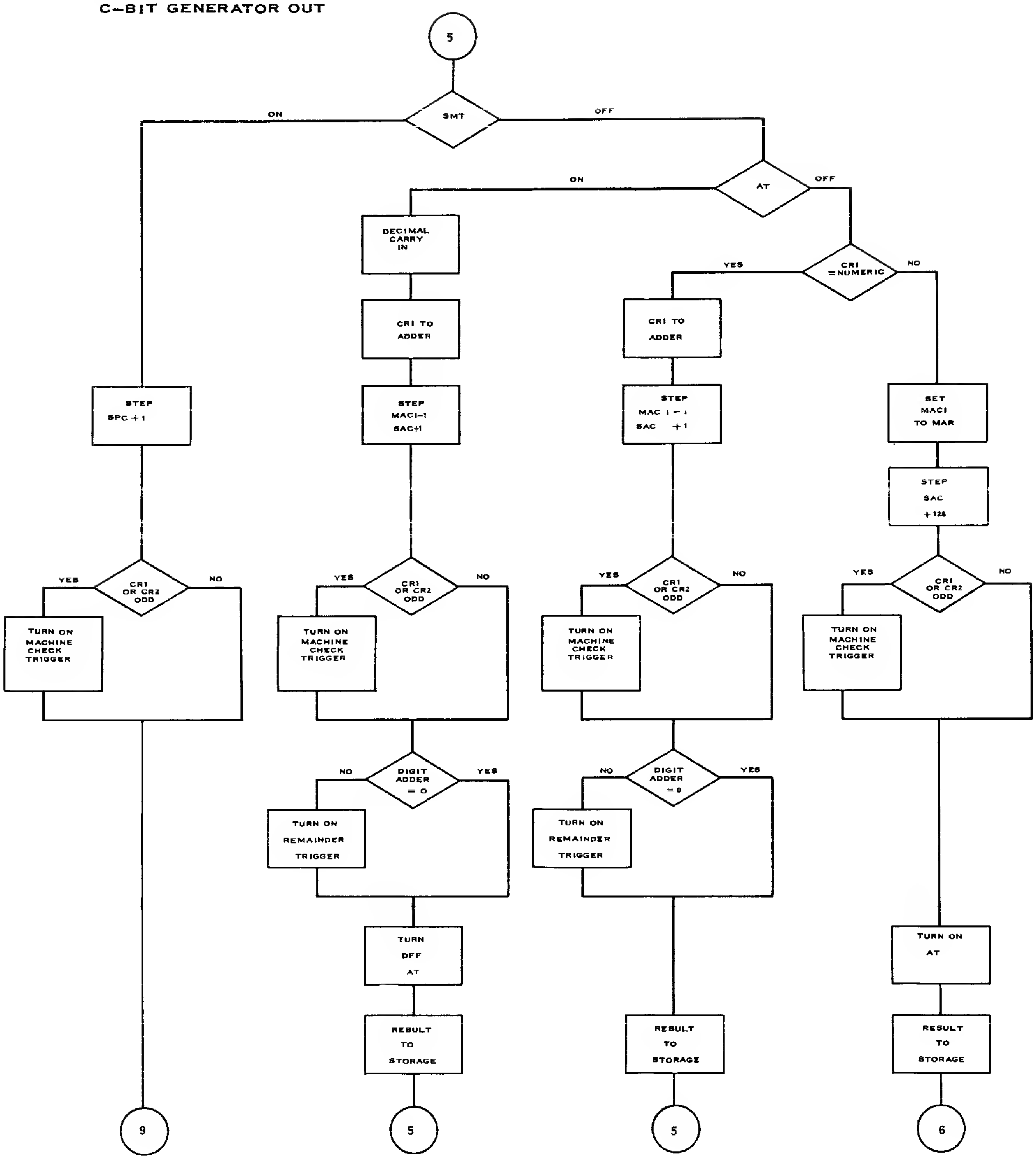


FIGURE 80

DIV

TYPE CYCLE VI
UNCONDITIONAL OPERATIONS:

READ STORAGE
SET "1" IN CR1
C–BIT GENERATOR OUT

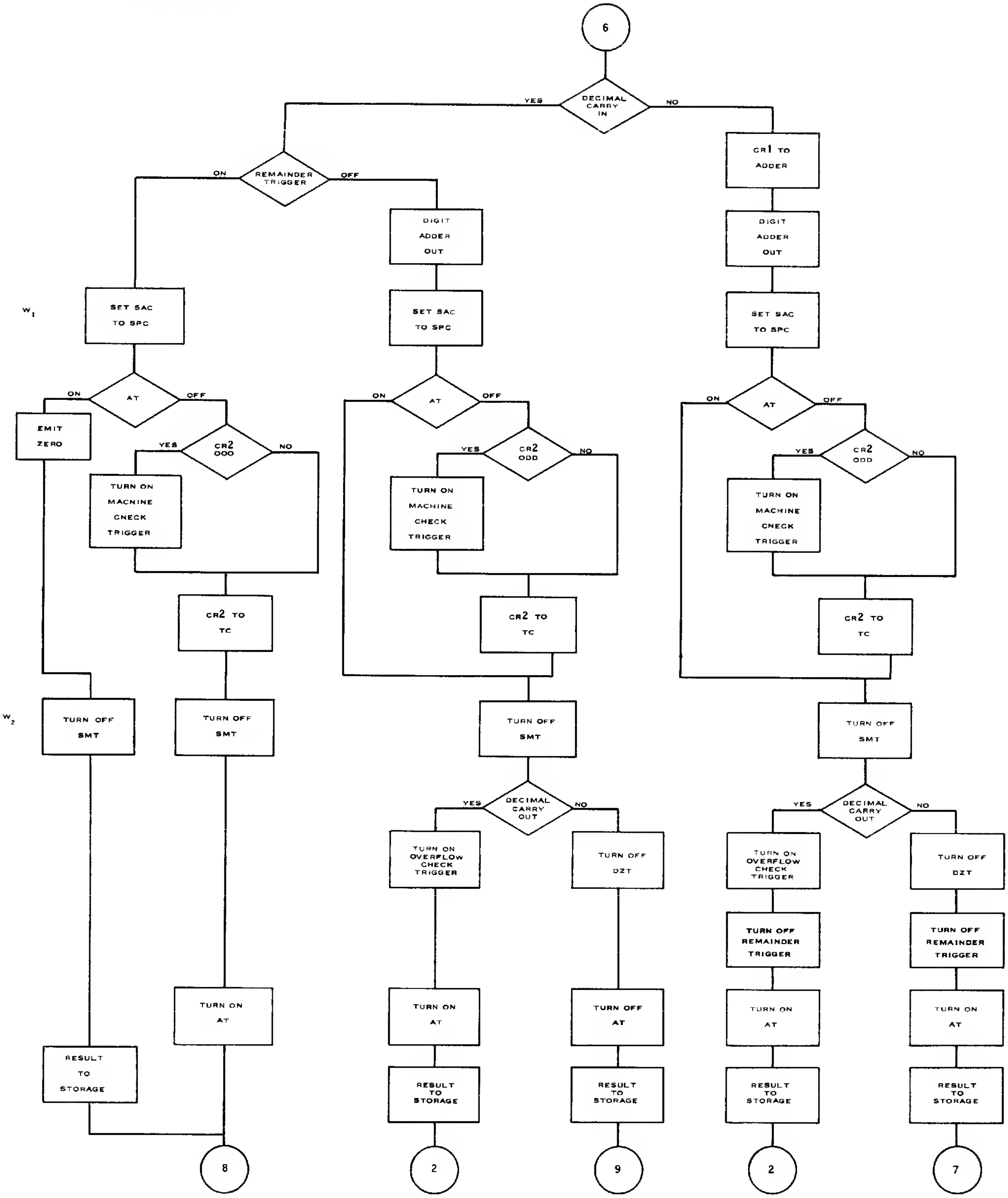


FIGURE 81

DIV

TYPE CYCLE VII
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CR2 TO TC
DIGIT ADDER OUT
C-BIT GENERATOR OUT
RESULT TO STORAGE

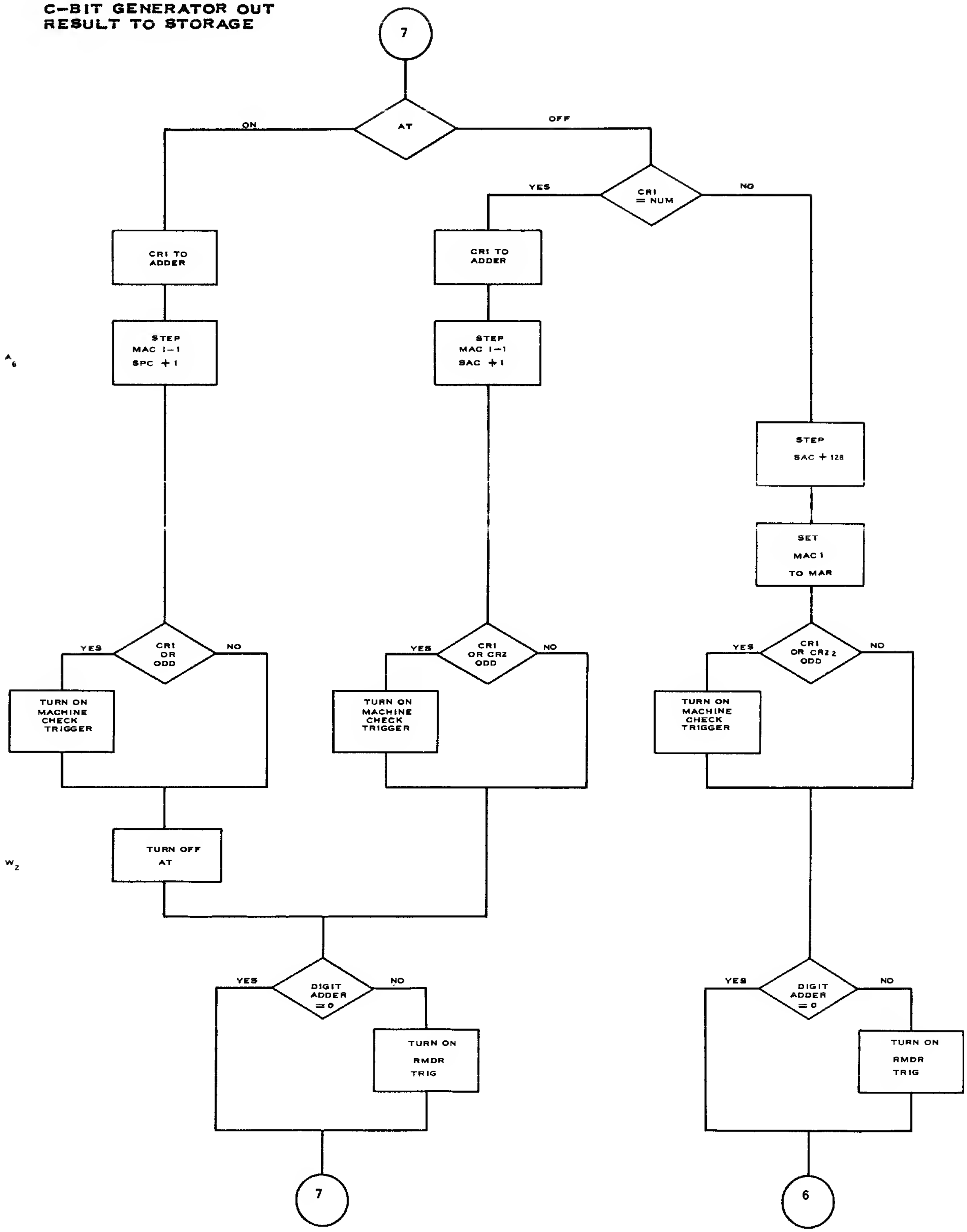


FIGURE 82

DIV

TYPE CYCLE VIII
UNCONDITIONAL OPERATIONS:

READ MEMORY
READ STORAGE
CR2 TO TC
COMPLEMENT
DIGIT ADDER OUT
C–BIT GENERATOR OUT
RESULT TO STORAGE

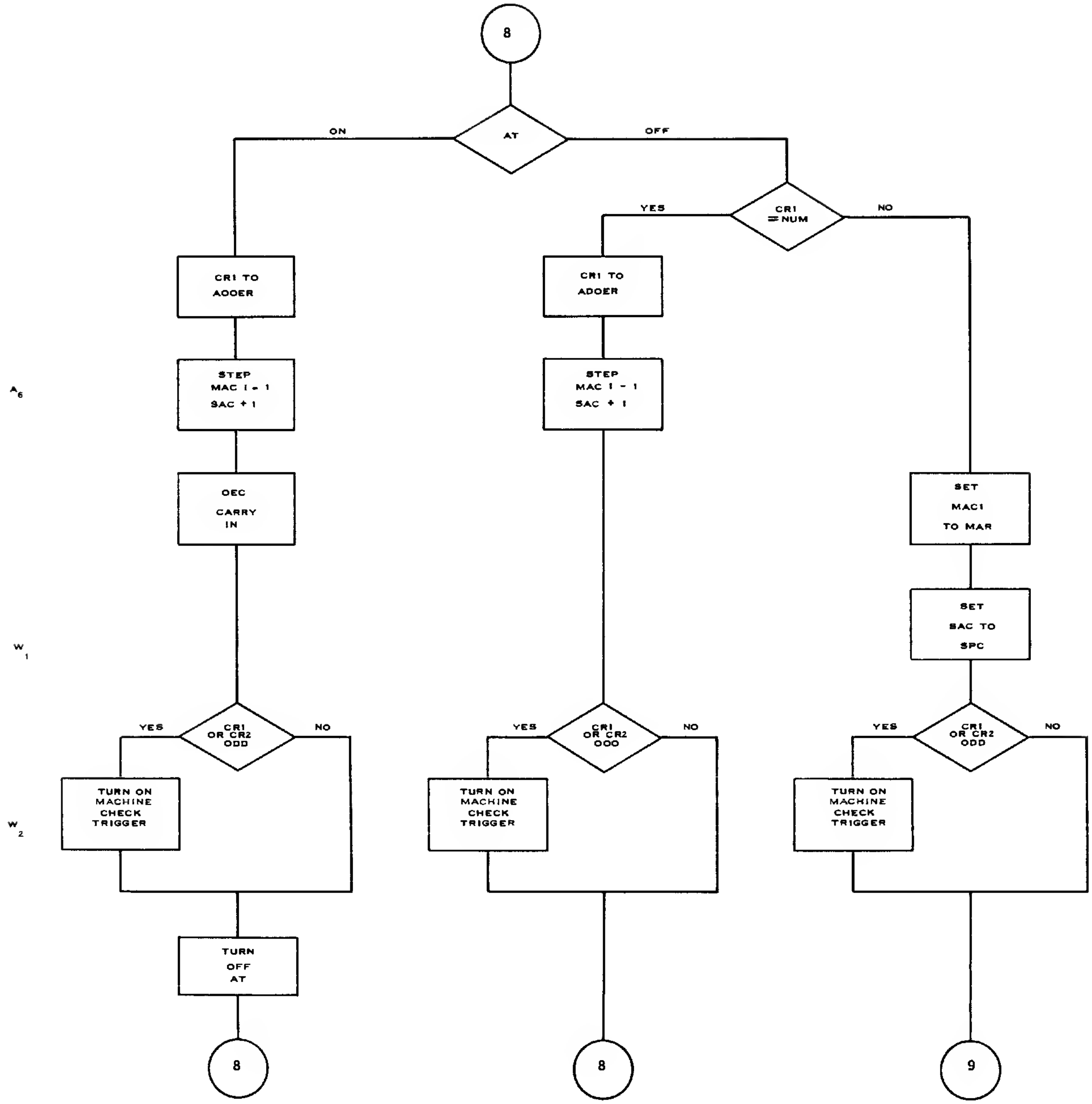


FIGURE 83

DIV

TYPE CYCLE IX

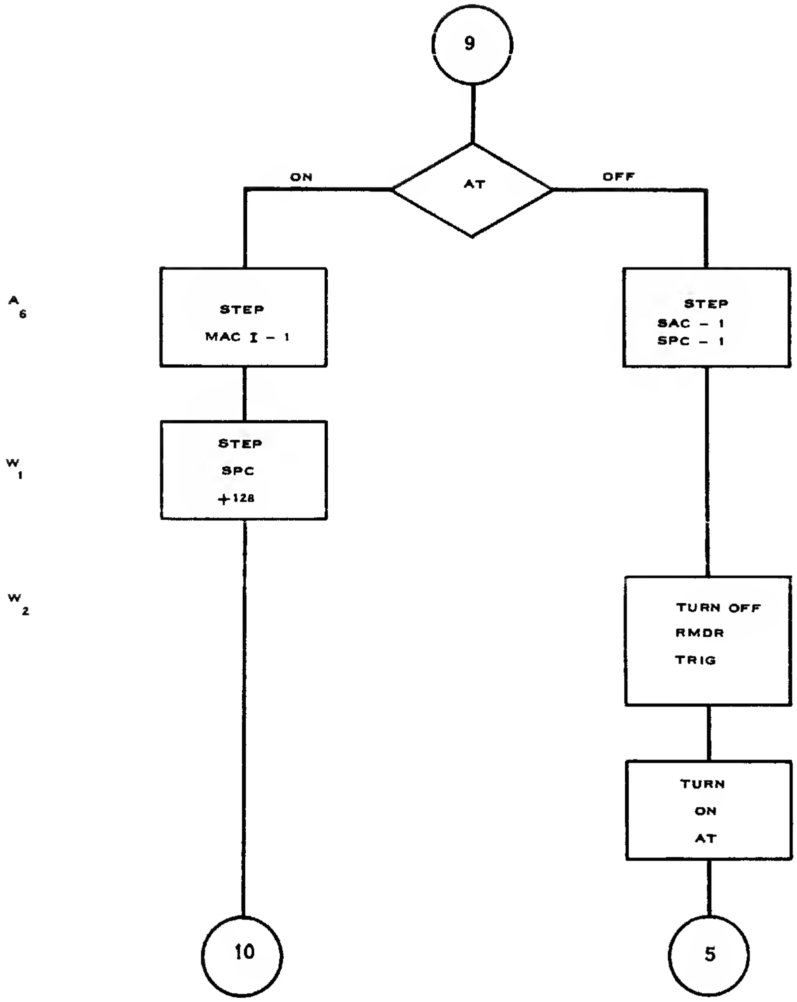


FIGURE 84

DIV

TYPE CYCLE X

READ MEMORY

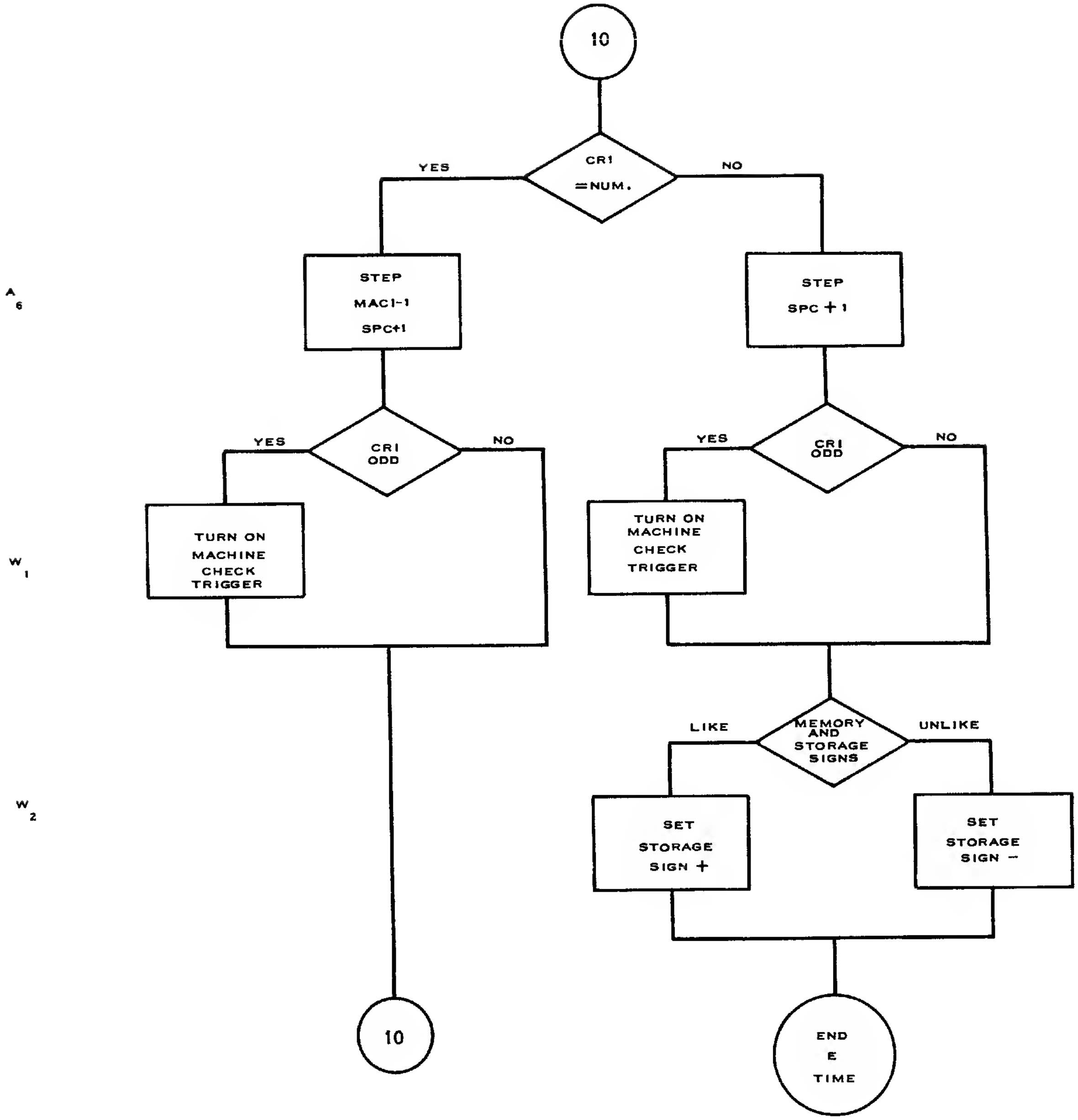


FIGURE 85

DIVISION EXAMPLE

Division Carried Out:
189 ÷ 9 = 20 with remainder: 6

Memory $\overset{+}{b9}$
SPC = Δ

Accumulator @186+
SAC = ↑

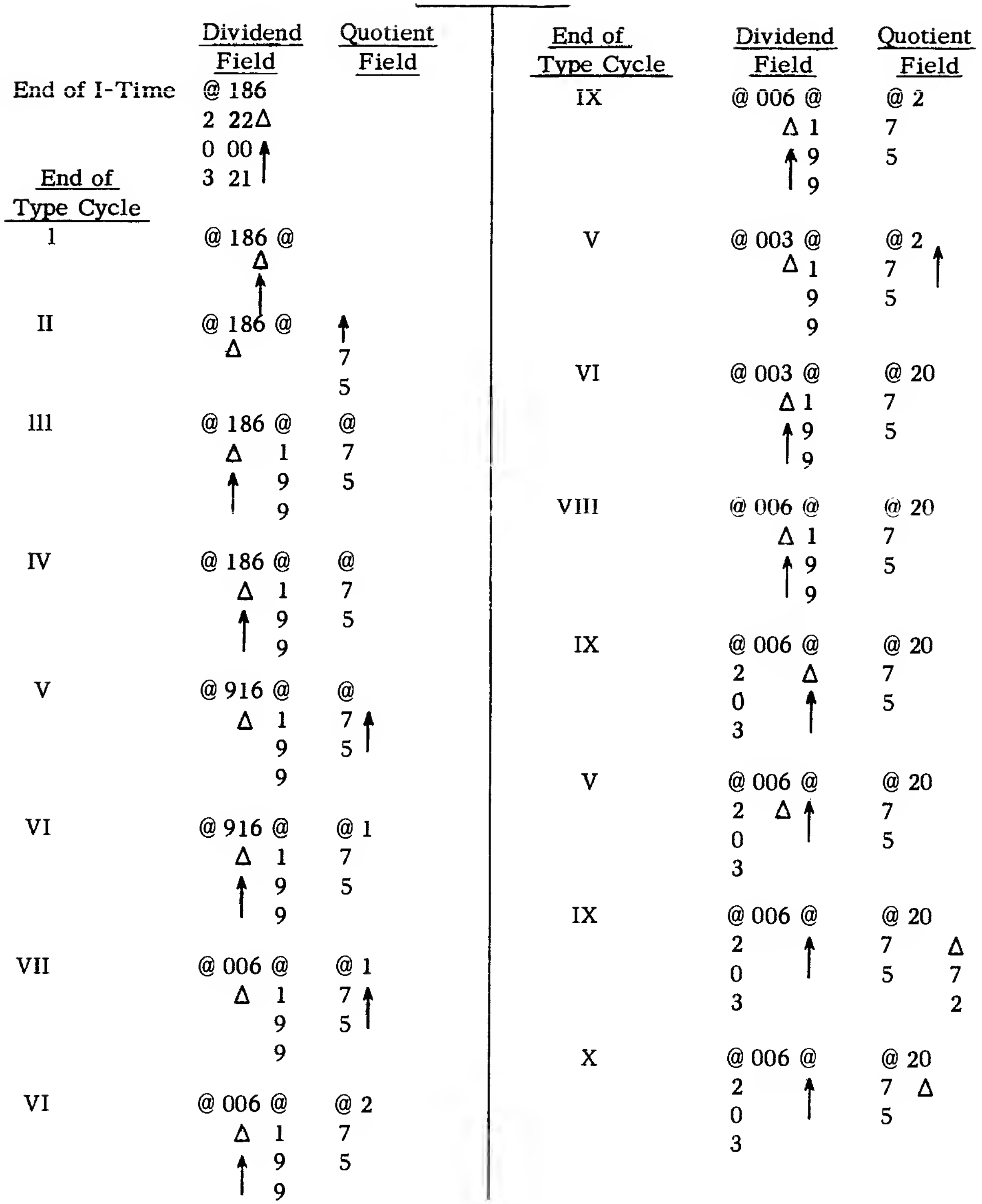


FIGURE 86

APPENDIX

True Complementer (TC)

The true complementer is used to obtain the 9's complement of a binary coded decimal digit. As explained previously in the case of the adder, the arithmetic operation, in this case complementation, is achieved through a sequence of logical circuitry such as And and Or gates and inverters (see Figure 87).

The digit to be complemented is, of course, represented by bits or no bits in the 1, 2, 4, and 8 channels. Let these channels be designated by N_1, N_2, N_4, and N_8, respectively, and their 9's complement by C_1, C_2, C_4, and C_8, respectively. Then Figure 87 clearly indicates:

1. C_1 will be in a bit condition if, and only if, N_1 is in a no bit condition.

2. C_2 will always be in the same condition as N_2.

3. C_4 will be in a bit condition if, and only if, either

   $$N_2 = 0 \text{ and } N_4 = 1$$

   or

   $$N_2 = 1 \text{ and } N_4 = 0$$

4. C_8 can only be in a bit condition if neither N_2, N_4 nor N_8 are in a bit condition.

The number zero appears in the adder circuits as the absence of all numerical bits. As the number zero passes from CR1 or CR2 to the adder circuits, the 8 and 2 bits are suppressed. As may be noted from Figure 87, the complement of a number without any numerical bits is 9.

As explained previously, the 10's complement of a number is created from the 9's complement by actuating a carry-in when obtaining the 9's complement of the units digit.

THE TRUE COMPLEMENTER

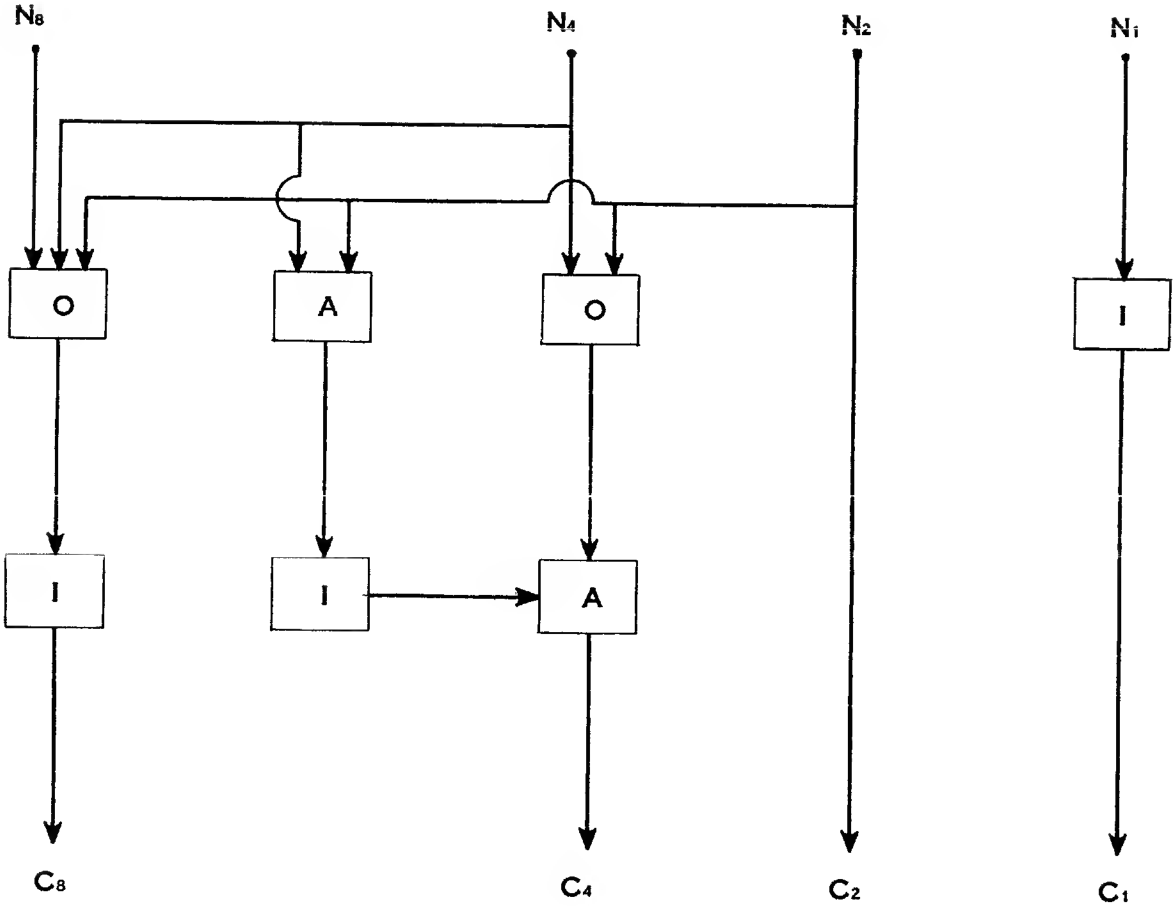


FIGURE 87

Abbreviations Used

ASU	Auxiliary Storage Unit
AT	Auxiliary Trigger
CE	Character Emitter
CPU	Central Processing Unit
CR1 or 2	Character Register 1 or 2
DA	Digit Adder
DZT	Digit Zero Trigger
ECC	Execution Character Cycle
E-Time	Execution Time
IC	Instruction Counter
ICC	Instruction Character Cycle
I/O	Input-Output
I-Time	Instruction Time
MAC I or II	Memory Address Counter I or II
MAR	Memory Address Register
MASR	Memory Address Select Register
MBR	Memory Buffer Register
MET	Memory End Trigger
MR	Multiply Register
RT	Remainder Trigger
SAC	Storage Address Counter
SASR	Storage Address Select Register
SBR	Storage Buffer Register
SMT	Storage Mark Trigger
SPC	Starting Point Counter
SSR	Storage Select Register
TC	True Complementer
ZA	Zone Adder

IBM®

International
Business Machines
Corporation

590 MADISON AVENUE
NEW YORK 22, N.Y.

32-7627 PRINTED IN U.S.A.